ARI Research Note 98-12

Cognitive and Motivational Consequences of Tutoring and Discovery Learning

Brian J. Reiser
Northwestern University

William A. Copen
Princeton University

Michael Ranney
University of California, Berkeley

Adnan Hamid
Princeton University

Daniel Y. Kimberg
Carnegie-Mellon University

Research and Advanced Concepts Office
Michael Drillings, Chief

June 1998

U.S. Army Research Institute
for the Behavioral and Social Sciences

U.S. Army Research Institute
for the Behavioral and Social Sciences

A Directorate of the U.S. Total Army Personnel Command

EDGAR M. JOHNSON
Director

Research accomplished under contract
for the Department of the Army

Northwestern University

Technical Review by

Joseph Psotka

NOTICES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1998	FINAL Aug92 - Mar94

4. TITLE AND SUBTITLE
Cognitive and Motivational Consequences of Tutoring and Discovery Learning

5. FUNDING NUMBERS
MDA 903-92-C-0114
PE 0601102A
20161102B74F
TA2901
WU C05

6. AUTHOR(S)
B. Reiser, W. Copen, M.Ranney, A. Hamid, D. Kimberg
(Northwestern Univesity)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
School of Education and Social Policy
Northwestern University
2115 North Campus Drive
Evanston, IL 60208-2610

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Institute ATTN: PERI-BR
5001 Eisenhower Avenue
Alexandria, VA 22333-5600

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
Research Note 98-12

11. SUPPLEMENTARY NOTES
COR: Drillings

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release; Distribution is Unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*:

A central controversy in the design of instruction concerns the amount of freedom or guidance that should be provided to students. We examined the cognitive and motivational consequences of guidance and freedom in a learning environment used by students learning introductory programming. Students worked with one of three interactive learning environments that varied in the amount of freedom to explore or guidance provided. We argue that discovery learning creates more opportunities for students to assess how well they can overcome obstacles, and their resulting attitudes toward their past and future success in the domain relies heavily on this type of attribution. The positive or negative nature of that attribution will depend on their relative success in achieving their goals.

14. SUBJECT TERMS
Learning , Instruction, Interactive learning, tutoring, discovery learning

15. NUMBER OF PAGES
83

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

Submitted, in review

COGNITIVE AND MOTIVATIONAL CONSEQUENCES OF TUTORING AND DISCOVERY LEARNING

Brian J. Reiser
Northwestern University

William A. Copen
Princeton University

Michael Ranney
University of California, Berkeley

Adnan Hamid
Princeton University

Daniel Y. Kimberg
Carnegie-Mellon University

Address correspondence to:

Brian J. Reiser
School of Education and Social Policy
Northwestern University
2115 North Campus Drive
Evanston, IL 60208-2610
Tel: 708-467-2205
Fax: 708-491-8999
Email: reiser@ils.nwu.edu

Abstract

A central controversy in the design of instruction concerns the amount of freedom or guidance that should be provided to students. We examined the cognitive and the motivational consequences of guidance and freedom in a learning environment used by students learning introductory programming. Students worked with one of three interactive learning environments that varied in the amount of freedom to explore or guidance provided. The *Guided* group solved a set of assigned problems worked with a tutoring system (GIL) that interrupts and provides explanatory feedback whenever students make mistakes. An *Exploratory* group worked on the same problems using an exploration-oriented version of the system that does not interrupt upon errors, but instead enables students to test ideas and receive feedback at their request. A *Free* group used the same exploratory learning system but was free to generate their own tasks. The tutorial guidance helped students solve the assigned problems more quickly, although there were no differences in the subjects' understanding of programming constructs or their ability to construct programs. However, both discovery learning groups were more effective in detecting bugs in programs, presumably due to their experiences finding and repairing their own errors. For students required to solve assigned problems, the positive or negative nature of the learning environment's motivational consequences appeared to depend upon the relative ability of the student. The learning environments differentially affected the attitudes of high and low ability students toward the domain and their assessments of the success of their performance. Low ability students in the discovery learning situation tended to exhibit more negative judgments about their performance and the computer's assistance than comparable students who received tutoring. In contrast, high ability students tended to exhibit somewhat more positive opinions of their performance in the discovery learning environment than comparable students in the tutoring situation. We argue that discovery learning creates more opportunities for students to assess how well they can overcome obstacles, and their resulting attitudes toward their past and future success in the domain relies heavily on this type of attribution. The positive or negative nature of that attribution will depend on their relative success in achieving their goals.

The Debate Over Discovery Learning

One of the most controversial issues in the design of instruction concerns the amount of freedom or guidance that a learning environment should provide. Advocates of learning by discovery have emphasized the student's active control of learning. In this view, students should be free to explore a domain, should learn by inferring principles rather than being told them, and should be free to make errors and learn by recovering from them (Bruner, 1961; Papert, 1980; Schank & Jona, 1991). In contrast, investigations of novice problem solving have suggested that errors can often be costly and lead to counterproductive floundering (e.g. Lewis & Anderson, 1985), leading some researchers to argue that providing careful guidance of students' problem solving is important in facilitating learning (Anderson, Boyle, & Reiser, 1985; Anderson & Corbett, 1993). This paper presents an empirical investigation of the effects of guidance and freedom to explore in a learning environment. First, we review the arguments for discovery learning and the concerns about this method. The experiment examines students learning LISP programming using three computer-based learning environments that vary in the guidance and freedom for exploration they provide. We consider both the cognitive and motivational outcomes of the guidance and the freedom to explore provided in these learning environments.

Proponents of Discovery Learning

A leading advocate of early interest in discovery learning was Jerome Bruner (1961, 1966). Bruner argued that learning in schools occurs typically in "expository mode," in which the teacher communicates information to the student and controls the structure and content of the interaction. In contrast, more efficacious learning occurs in "hypothetical mode," in which the teacher and learner are in a cooperative relationship, and the student plays a larger role in determining the nature of the interaction. The student can actively evaluate incoming information and spontaneously form new hypotheses about the topic of the learning. Bruner argued that the instruction process is more meaningful to learners who have a significant role in determining its course. Early investigations of the learning by discovery approach occurred in domains such as physics, in which students controlled the gathering of data to test their hypotheses provoked by classroom demonstrations (Suchman, 1961), and mathematics, in which students worked through examples and tried to

discover mathematical principles (Davis, 1966).

Much of the recent interest in the use of computers in education has centered on using the computer to provide an environment that students can explore under their own control. Debates about the best use of computers in instruction have rekindled interest in the issue of learning by controlling one's own investigations of a domain. Proponents of discovery learning environments have focused on the process of hypothesis generation and testing. Schwartz (1989) argued that educational computer programs should be "intellectual mirrors," enabling students to investigate a domain while providing little control of the students' interactions. For example, in the Geometric Supposer system (Schwartz & Yerushalmy, 1987; Yerushalmy, Chazan, & Gordon, 1990), students are encouraged to be creative in making conjectures about geometry and then to test them by having the computer construct geometric objects with the desired properties. Schwartz argued that learning environments that attempt to monitor students' problem solving to control the instructional process would only complicate and frustrate learning. Dugdale (1982) and Schoenfeld (1988) have designed learning environments in which students learn by making predictions about mathematical behavior and testing them through the program's graphical representation of the students' mathematical expression. Papert's (1980) arguments for the pedagogical benefits for children learning Logo include the freedom Logo provides students to explore the consequences of their actions and set their own goals to investigate the computer's responses, rather than be restricted to working on predefined problems.

Recently, some instructional systems have been designed to capitalize on the process of learning through making predictions by providing facilities for students to conduct experiments. An important trend in learning environment design is to construct microworlds that simulate essential properties of a domain which students can explore. For example, students can use Smithtown to learn about economics by varying aspects of the current economic situation in a small town (such as prices and availability of particular goods) and observing the effects on prices and amount of goods sold (Shute & Glaser, 1990; Shute, Glaser, & Raghavan, 1989). Students using Voltaville can learn about electric circuits by conducting experiments in the simulation environment to explore the concepts of voltage, current and resistance (Schauble, Glaser, Raghavan, & Reiner, 1991). Grade school students working with ThinkerTools can perform simple experiments in a microworld to explore laws of

mechanics (White, 1993).

In summary, in a discovery learning context, the student is in primary control of the interaction. The student's goals and interests determine the learning agenda. Students are encouraged to form hypotheses, test these hypotheses, and observe the results. Students learn by inferring principles from the gathered observations, rather than by the teacher communicating the central principles. This situation contrasts with more didactic and constraining teaching strategies in which an instructor presents ideas that have been included in a preestablished curriculum designed according to the logical structure of the domain of study.

The arguments for discovery learning have included claims for both cognitive and motivational benefits. The proposed cognitive benefits are that students learn better or learn more general skills when learning by discovery. The proposed motivational benefits are that students find a domain more intrinsically motivating when learning through exploration, and that they will develop more positive attitudes toward the domain and toward their abilities. Next we briefly review the arguments and evidence bearing on these issues.

Proposed Cognitive Benefits of Discovery Learning

The early arguments for discovery learning claimed that knowledge acquired through this method was better understood, better organized, and more amenable to future use (Bruner, 1961; Suchman, 1961). Yet early empirical investigations of discovery learning yielded mixed results. Some studies comparing directed instruction and learning by discovery found that directed instruction led to superior learning, while others found an advantage for the discovery students (Wittrock, 1966). However, these studies have been criticized for the use of rote learning as the control condition for comparison with discovery learning (Ausubel, 1963; Cronbach, 1966; Resnick & Ford, 1981). These studies confounded the directed versus exploratory nature of the learning with the content of the learned material. Advantages for the discovery group may have arisen because the learning situation led students to focus on the structure of the problems and formulate principles, while the rote learning group had no motivation and in many cases no opportunities to learn organizing principles for the material. Hence these studies were potentially measuring the advantages of meaningful learning over rote learning, rather than any advantage of discovery

learning over direct instruction. Wittrock (1966) argued that the discovery learning claims were in reality a number of related claims concerning meaningfulness, induction, student control, and so on, and that these claims should be evaluated individually rather than by simple comparisons of discovery learning with other learning situations. In recent work, the arguments have become more focused and potentially amenable to experimental investigation. In this section we review the particular claims for discovery learning and summarize the empirical evidence.

The first type of argument for learning by discovery relies on the general principle that a greater amount of self-generated processing results in superior memory and performance of the target skill than a more passive encoding method. Recent theories of learning argue for the importance of "learning by doing" (Anderson, 1983; Laird, Newell, & Rosenbloom, 1987). For example, it is generally considered better for students to solve problems themselves than merely to observe solutions performed by a teacher or to read worked-out solutions in a textbook. Analyses of students learning in domains such as mathematics and computer programming suggest that acquiring skills in a new domain requires that students attempt to apply the material presented in instructional text to solve new problems (Anderson, 1987; Trafton & Reiser, 1993; VanLehn, 1990). Extending this superiority of learning through active problem solving over passive observation, it might be argued that students who construct or discover a procedure for themselves would acquire more elaborated or robust knowledge than students who were guided in their problem solving.

Arguments for the robustness of knowledge acquired through discovery have been made by Carroll and his colleagues (Carroll, Mack, Lewis, Grischkowsky, & Robertson, 1985). They argued that instructional texts should encourage "active learning" by providing information to enable students to explore but never explicitly laying out procedures to be memorized. They found that students given a "Minimal Manual" that encouraged exploration learned more about operating a word processor than students given a tutorial manual that led them step by step through a series of examples. These results suggest the importance of eliciting reasoning from the student in contrast to asking students to learn procedures from detailed step-by-step instructions. Indeed, the degree to which students explain instructional examples to themselves greatly influences how much is learned from them (Chi, Bassok, Lewis, Reimann, & Glaser, 1989). Students who treat instructional examples as problems

to be solved and attempt to explain each step learn more from these examples than students who treat them as text to be read and understood. Such elaborated representations of solutions produced by self-explanation or by solving problems may contain representations of a problem's goal structure and records of obstacles and decisions during reasoning, and are more useful in guiding later problem solving than more passively encoded examples (Carbonell, 1986; Trafton & Reiser, 1993; VanLehn, Jones, & Chi, 1992).

Carroll et al. (1985) argued that their results support the effectiveness of learning by discovery. However, it is difficult in these studies to disentangle the effects of exploration from the effects of problem solving. Charney and Reder (1986) suggested that it may be the problem solving nature of the task, in which students must select a step to take, carry it out, and evaluate the consequences, that accounts for the better learning in the Carroll et al. experiment, and not the exploratory feature of the problem solving. Charney and Reder found that students learned a word processor more effectively by solving assigned problems than students who only studied the same examples or students in a "guided practice" group who were told exactly what actions to take and hence did not have to actively construct solution plans themselves. Thus, although the problem solving subjects worked on assigned problems and were not free to explore or set their own tasks, they benefited just did as students who learned by discovery in the investigations of Carroll et al. This finding is consistent with Ausubel's (1963) argument that the benefits documented for discovery learning in early studies were due not to the method of learning, but to the fact that these students learned meaningful principles not available to students in other learning conditions.

Taken together, these results suggest the importance of engaging the student in the reasoning required to produce solutions. They suggest the pedagogical effectiveness of requiring students to learn by solving problems. It is possible that it will be even more effective to let students explore the space by themselves, construct their own problems and discover their own solution methods, but the studies by Carroll et al. and earlier studies reviewed by Wittrock (1966) do not separately test these benefits of exploratory learning.

A second argument for discovery learning concerns the central role of learning from errors. Proponents of discovery learning argue that an important benefit of exploration is the potential for encountering unanticipated events and then learning

to reproduce unanticipated positive outcomes or avoid negative outcomes. Schank and his colleagues proposed that students learn when their expectations fail and they attempt to explain the unanticipated success or failure (Schank, 1986; Schank & Leake, 1989). In this view, a learning environment should encourage students to ask questions, to predict answers, to be wrong, and to explain and thereby profit from mistakes (Schank, 1990; Schank & Edelson, 1990; Schank & Jona, 1991). In contrast, didactic teaching strategies aim for the elimination of expectation failures, and according to this view cannot support the development of creative thinking and flexible problem solving.

Indeed, the use of failed expectations as opportunities for instruction can be observed in the behavior of expert teachers. Collins and Stevens (1982) found that teachers employing inquiry-based strategies utilize an entrapment strategy, in which the teacher leads students into articulating a prediction which they then discover is incorrect. Students are then forced to actively analyze the prediction in order to determine why it was wrong. A similar strategy, called "torpedoing," was used by Davis (1966), in which students are given examples, asked to form generalizations to explain the examples, and then given new examples that reveal a flaw in the generalization, thus pushing the students to refine and extend their explanations to cover the new cases.

In evaluating these arguments concerning the active nature of the problem solving and the utility of learning from errors, it is important to consider the potential costs of problem solving as well. Although learning by solving problems is clearly effective, in some situations students may learn more by studying examples than by unguided problem solving (Sweller, 1988; Sweller & Cooper, 1985). Sweller and his colleagues argued that if problem solving forces students to rely upon weak methods such as means-ends analysis, this can interfere with acquiring the problem schemata necessary to gain proficiency in the skill, and therefore learning is more effectively accomplished by understanding a series of worked-out examples instead. In some domains, if a problem's solution was obtained with excessive floundering, it may be difficult for students to remember how they constructed the solution and hence learning from that solution will be difficult (Lewis & Anderson, 1985). Anderson et al. (1985) argued that when errors in a solution lead to a costly floundering episode in an attempt to recover, such episodes interfere with learning rather than produce desirable learning outcomes. Anderson and Corbett (1993) argued that in

domains such as mathematics and programming, in which completed solutions provide a record of the successful problem solving decisions, learning will be a function of the solutions students achieve and not strongly affected by the trajectories they take to reach the same solutions. In these types of domains, leaving students free to explore may lengthen learning time with no increase in amount learned (Anderson, Conrad, & Corbett, 1993). Hence, they argued that a learning environment should minimize floundering time in problem solving.

The third argument for discovery learning states that this type of learning situation helps students learn to control their own learning. Theorists have argued that students need to acquire "a working heuristic of discovery" (Bruner, 1961) such as general scientific inquiry skills (Shute & Glaser, 1990; Shute et al., 1989) and error management skills that are necessary to systematically learn new domains. For example, the ThinkerTools environment is designed so that students learn not only simple laws of mechanics but also more general scientific experimentation strategies (White, 1993). Bereiter and Scardamalia proposed that learning environments should promote "intentional learning" in which students learn how to set goals, diagnose their own errors, and monitor and organize their knowledge (Bereiter & Scardamalia, 1989; Scardamalia, Bereiter, McLean, Swallow, & Woodruff, 1989).

Research on learning general problem solving heuristics through discovery has provided a mixture of evidence. One reason for concern about students learning through exploration is that naive strategies for hypothesis testing often exhibit a bias to seek evidence to confirm hypotheses rather than discriminate between competing theories (Klayman & Ha, 1987; Nisbett & Ross, 1980) and a bias to accept confirming evidence while seeking alternative explanations for disconfirming evidence (Nisbett & Ross, 1980). Examination of students' behavior in exploratory learning situations reveals that there are indeed inquiry skills important for learning. Shute et al. (1989) investigated students' experimentation strategies in their discovery microworld for economics. Shute et al. found that students differed in the degree to which they utilized effective heuristics for constructing tests of hypotheses, and those students exhibiting better heuristics also learned more about economics as a result of their inquiries. Similarly, Schauble et al. (1991) found strong differences in what students learned from their exploration in an electrical circuit simulation environment based on the sophistication of their prior causal models about the domain. Klahr, Dunbar, and Fay (1990) found that students'

generation and pursuit of hypotheses was markedly influenced by their prior beliefs about the phenomenon. This line of research indicates that students who come to the task with more sophisticated prior knowledge and with more effective hypothesis generation, experimentation, and data organization skills learn more from their experimentation. Therefore, students may be unlikely to collect evidence through their explorations that is sufficient for learning the desired concepts. Thus, there is evidence for the importance of effective inquiry heuristics in learning. However, it remains unclear how these heuristics are actually learned, and there is little evidence that experience with discovery learning improves these heuristics. Indeed, although there may be clear advantages to acquiring discovery or inquiry heuristics, it may be that these heuristics are not best learned by discovery (Wittrock, 1966).

One promising area to look for evidence that discovery learning leads to inquiry and other general skills would appear to be the research on Logo programming, which was touted as an example environment in which students will acquire not only computer programming constructs but also general problem solving skills through their exploration (Papert, 1980). However, recent evaluations of the Logo claims typically have failed to demonstrate that students learning Logo acquire general problem solving heuristics such as means-ends analysis, forward chaining, or other effective planning strategies (Pea, Kurland, & Hawkins, 1987). In fact, Logo can be an effective environment in which to learn planning or debugging strategies, but these outcomes have not been demonstrated when students are left free to explore, but only when these strategies were an explicit topic of the teacher's instruction, and this required a fair amount of teacher intervention to model the target reasoning process (Carver, 1988; Swan, 1989).

Another potential problem with discovery learning is the danger that students will fail to discover important aspects of the domain. Ausubel (1963) argued that while the idea of finding out for oneself may sound appealing, in practice it may often be the case that people fail to uncover important principles. An informed teacher can perform more effective instruction through explicit communication of insights and ideas critical for meaningful understanding of the domain. Research on children learning Logo provides some support for this concern — students who are left to learn purely through their own exploration may fail to acquire important computer programming skills. For example, Mayer (1985) found that students who learned by discovering BASIC on a computer developed a greater number of fundamen-

tal misconceptions than did students who learned exclusively from a text. Dalbey and Linn (1985) reported that students often fail to learn Logo well when they are left on their own. Similarly, Kurland, Clement, Mayby and Pea (1987) found that students learning Logo in a discovery curriculum only learned the rudiments of programming constructs such as variables and procedures but did not exhibit an understanding of the flow of control of programs or the structure of the language required for "sophisticated understanding" such as passing variables between procedures or reusing procedures across programs. Although these studies did not directly compare students learning Logo through discovery to those learning with more directed instruction, the results do suggest that learning through exploration may sometimes fail to achieve the proficiency with the domain that may be desired.

In summary, the claimed cognitive benefits for learning through discovery concern a more elaborated representation of problem solving skills, more effective learning from errors, and general self-monitoring and inquiry skills. Although the importance of inquiry skills has been demonstrated, there is little evidence to date that learning through discovery leads to acquisition of effective general problem solving heuristics. Two general concerns with discovery learning have been raised. The stronger claim is that any benefits of discovery are outweighed by the advantages of a good teacher, i.e., learning is more effective with lessons planned by a teacher according to the conceptual structure of the domain and with guidance provided to keep problem solving productive. The second concern is that while discovery may have some benefits, students who are left to explore may never uncover some important principles if they are not included in a preestablished curriculum.

Proposed Motivational Benefits of Discovery Learning

The second important class of claims for discovery learning emphasizes motivational effects. The arguments for the motivational benefits of discovery learning state that knowledge acquired through discovery is more valued, and students develop more confidence in themselves and more positive attitudes toward the domain (Bruner, 1961; Suchman, 1961). Bruner (1961) argued that learning by discovery is driven by the rewards of the task itself, and that instruction should be designed to "enlist spontaneous learning" rather than to drive learning by extrinsic rewards (Bruner, 1966). These early advocates, however, did not propose a model of how these

motivational benefits arise from a learning situation.

More recently, Lepper and his colleagues have developed a theory of intrinsic motivation in learning (Lepper & Chabay, 1985; Lepper & Malone, 1987; Malone & Lepper, 1987). Malone and Lepper (1987) considered aspects of instructional situations likely to lead to interest in the domain for the reward of engaging in the activity itself. Malone and Lepper's analyses of motivational influences on individual learners provide a framework in which we can evaluate the potential motivational consequences of discovery learning. We briefly review these analyses and consider related claims and evidence.

The first characteristic of learning situations considered by Malone and Lepper (1987) is an optimal level of "challenge." They argued that challenge should make the task interesting and effortful without being frustrating. To accomplish this, the activity should either present clear, fixed goals or allow learners to set the goals themselves. In addition, there should be an uncertainty of outcomes, frequent and helpful instructional feedback (so students can adjust their goals during learning), and performance feedback that promotes self-esteem. Challenge is an interesting issue for evaluating guided and discovery based learning. It seems reasonable that either extreme in a learning environment may lead to negative motivational outcomes. If the student receives too much help, then the learning activity will not be sufficiently challenging. However, if students are left free to explore and achieve goals without guidance, a difficult task may become too challenging, leading to negative self-evaluations.

The second intrinsically motivating aspect of a learning activity considered by Malone and Lepper (1987) is curiosity. Intrinsically motivating learning activities are those that engage the learner's curiosity by providing discrepancies between the learner's expectations and actual events. Thus, the types of failures of expectations that Schank and Jona (1991) present as the key opportunities for learning are those that lead to increased interest in the topic of study in Malone and Lepper's model. In addition, the learning activity can promote curiosity by exposing incompleteness or inconsistency in the learner's knowledge. Thus, students who realize that their knowledge is inconsistent or incomplete when arguing with inquiry teachers (Collins & Stevens, 1982) or when causing unexpected events through exploration (e.g. Ranney & Thagard, 1988; White, 1993) are likely to become more curious about the domain and motivated to investigate further.

The third aspect of intrinsic motivation discussed by Malone and Lepper (1987) is control. They argued that students should feel in control of their actions and the responses of the environment. The environment should provide a moderately high level of choice so that the student feels in control of the direction of the interaction, but not confused and unsure of where to begin. This type of intrinsic motivation is clearly associated with discovery learning, in which the student controls the learning agenda, the topics and methods of exploration. Malone and Lepper also discussed the benefits of giving students the power to create large and interesting effects. One of the purported benefits of Logo is that it enables children to produce spectacular visual displays with only a moderate investment of effort (Papert, 1980), and recent experiments with Logo go even further in allowing students to program the behavior of physical objects (Resnick & Ocko, 1991).

Investigators have argued that learning driven by intrinsic motivation or "learning goals" is not only more enjoyable for the learner, but also leads to more effective learning than learning driven by "performance goals" (Dweck, 1986; Lepper, 1988). Intrinsically motivated students engage in the learning task because they find it rewarding, but extrinsically motivated students commit effort only to obtain an external reward, and may therefore devote less time and effort to learning activities (Lepper, 1988). In addition, intrinsic motivation may elicit more productive cognitive processing, leading to more efficient or more elaborative learning (Lepper, 1988). Extrinsically motivated students select less difficult and challenging problems, while intrinsically motivated students are more likely to select problems that will challenge their capabilities (Dweck, 1986; Pittman, Emery, & Boggiano, 1982). Similarly, students with learning goals tend to increase effort or modify their strategies in response to obstacles, in contrast to the less productive responses of students with performance goals such as withdrawal (Dweck, 1986). Consistent with these proposals, Nolen (1988) showed that motivational factors affect the learning strategies that students value and utilize. Students motivated by intrinsic properties of a learning task were more likely to value and employ deep-processing strategies (e.g., discriminate between important and unimportant information, integrate new information with old information, etc.), while students motivated by extrinsic concerns were more likely to value and employ surface-level strategies (e.g., repeatedly read an entire passage, memorize exact wording, rehearse salient facts). Because deeper processing strategies are more likely to lead to understanding and reten-

tion (Brown, Bransford, Ferrara, & Campione, 1983), these results suggest that the intrinsically motivated students will learn more from their studying time than extrinsically motivated students. Recent studies by Lepper and his colleagues suggest that "motivationally-enhanced" learning environments, in which a rather dry learning task is enhanced by setting it within a motivating fantasy context, can lead to superior learning outcomes (Lepper & Cordova, 1992; Parker & Lepper, 1992).

These analyses of the motivational issues involved in learning suggest the potential advantages of discovery learning environments. Learning through exploration certainly can provide more challenge than learning controlled by a teacher. The student's curiosity is invoked and can drive the particular problems or tasks set by the student. The control clearly belongs to the student rather than to an authority such as a teacher, or to a machine controlling the learning (as in "drill and practice" computer-based instruction). However, there may also be dangers lurking in these potential advantages. If the domain proves more difficult than the student's capabilities and most of the student's goals meet with failure, then the learning activity can lead to decreased motivation. Students' curiosities will not be satisfied if the students cannot achieve goals to find answers, resolve inconsistencies, and explain unexpected results. Such evidence of failure, if attributed by the student to a lack of ability, can have negative consequences for self-esteem, expectations, and future effort (Dweck, 1975; Weiner, 1979, 1985).

This review of evidence presents an ambiguous picture of the cognitive and motivational consequences of discovery learning. The evidence for superior learning outcomes acquired through discovery is mixed, and the observed positive effects may be due to researchers contrasting problem solving situations with more passive learning methods. The effectiveness of students learning from errors is largely undemonstrated, and there are reasons for concern with learning episodes consisting predominantly of errors and error recovery. The success of learning general inquiry and error management skills through discovery is also in doubt. Overall, there is little direct evidence for the cognitive benefits of discovery learning. Among the motivational consequences, we have reviewed both the reasons to expect that some characteristics of discovery learning situations lead to more intrinsic motivation, and reasons to expect that greater intrinsic motivation positively affects learning. However, there have been few investigations directly examining the motivational consequences of freedom or guidance in instruction.

An Empirical Investigation of Tutoring and Discovery Learning

The present study was designed to investigate both the cognitive and motivational consequences of guidance and freedom to explore in a learning situation. Certainly there is much middle ground possible between a pure discovery learning system and one in which students' problem solving is carefully guided. For example, some designers have explored mixed systems in which students have a chance to explore but can receive coaching upon request or upon strongly demonstrated need (e.g. Burton & Brown, 1982; Elsom-Cook, 1990; Lesgold, Lajoie, Bunzo, & Eggan, 1992). Other researchers have stressed the role of students working together to learn from their interactions with a microworld (e.g. Roschelle, 1992). However, there have been few investigations that directly examine the learning and the motivational consequences of providing guidance or freedom in a learning environment. Informed design of learning environments requires a better understanding of what the potential benefits and the costs are of providing coaching or of providing the freedom to explore. The present experiment is designed to begin to provide such evidence.

Intelligent tutoring systems provide a promising workbench for investigating these learning issues. An instructional system that can understand students' reasoning as they construct a solution is capable of providing careful guidance during problem solving (Anderson et al., 1985; Lesgold et al., 1992). A learning environment can be constructed to allow students the freedom to explore and work on the same problems without the constraints of the tutor's guidance. This provides the opportunity to keep most aspects of the target knowledge in the domain constant and keep the problem solving representations used by the student constant, while varying the amount of guidance and freedom to explore provided during learning.

We investigated these issues using three computer-based instructional systems that embody a range of pedagogical strategies, from constraining and didactic to discovery learning. The experiment examines programming novices learning to write LISP programs. All students read the same textbook and worked with one of three interactive learning environments. We evaluated the cognitive consequences (what the students learn) and the motivational consequences (the students' attitudes toward the domain and toward themselves).

Learning Environments Used in the Study

The three interactive learning environments cover a range from interventionist tutoring system to a free open-ended discovery learning environment. All three programs are based on GIL, *Graphical Instruction in LISP*. GIL is an intelligent tutoring system designed to help students understand how programs work as they construct and debug LISP programs (Merrill, Reiser, Beekelaar, & Hamid, 1992; Ranney & Reiser, 1989; Reiser, Beekelaar, Tyle, & Merrill, 1991; Reiser, Kimberg, Lovett, & Ranney, 1992; Reiser, Ranney, Lovett, & Kimberg, 1989). There are two characteristics of GIL that contribute to its pedagogical effectiveness. First, GIL is designed to be more congruent with effective problem solving processes. Second, GIL provides explanatory feedback as students work on problems. GIL understands a student's solution as it is constructed and hence it can provide feedback to warn the student about misconceptions or slips and provide strategic hints to assist the student's problem solving. By varying when GIL provides feedback and varying the degree of freedom students are given to explore, we can investigate the learning effects of these instructional designs. In this section we will present a brief summary of GIL and describe the guided learning and discovery learning environments that we have constructed for these investigations. GIL's problem solver, explanatory component, and graphical representation are more completely described by Reiser et al. (1992).

The first goal in GIL's design is to provide a *reasoning-congruent learning environment* to help scaffold effective problem solving (Merrill & Reiser, 1993; Merrill et al., 1992; Reiser et al., 1991; Reiser, Friedmann, Gevins, Kimberg, Ranney, & Romero, 1988). A difficulty in learning to solve problems in many domains is that the syntax of a problem's solution does not reflect the reasoning process required to construct the solution (Collins & Brown, 1988). A reasoning-congruent learning environment is designed to provide a better fit with students' reasoning, thereby minimizing the translation between their plans and the recording of inferences in the external record of their solution. GIL uses a graphical interface to help organize problem solving and make hidden information (such as data flow) explicit. GIL students build a program by connecting together objects representing program constructs into a graph, rather than by defining LISP functions in their traditional text form. In the early curriculum, students select a LISP function and specify its input and output data, thus making explicit assertions about the changing state of the

program's data. (In the later curriculum, not used in the present study, students are not asked to specify input and output data, but can obtain this information on demand.) A completed program in GIL consists of a graph specifying how a chain of LISP functions transforms the input data to achieve a particular type of output. The use of intermediate products makes the process of embedding functions to sequence operations on data more explicit, and is designed to lead to a better understanding of functions. The interaction allows the tutor to monitor students more closely and to provide more useful assistance as they learn programming plans than in an environment in which students specify only the final surface form of the solution.

In GIL's graphical representation, the structure of the solution being constructed mirrors the planning required to construct a program better than in a text-based representation. Reasoning chains are represented by branches of the graph joined together to achieve the final goal. Intermediate reasoning products are made explicit, so that students see the data manipulated as they construct the program. This helps students understand how particular algorithms work and, more generally, helps them learn the logic of embedding functions within other functions to construct an algorithm. Finally, the GIL interface enables students to plan in a variety of directions. Students can reason forward from the given data toward the goal, or can use goal decomposition and work backward from the goal toward the given data. The interface supports both types of reasoning and provides a distinct visual representation that mirrors the direction of reasoning. This facilitates students' reasoning about programs, because they often do not construct the components of solutions in the order in which these appear in the final surface form of the solution (Trafton & Reiser, 1991).

The three environments based on GIL are summarized in Table 1. The critical hypotheses in this study concern the cognitive and the motivational consequences of allowing students freedom to explore or providing tutorial guidance as they work to master a curriculum. Thus, the central comparisons in the experiment concern the Guided and Exploratory environments, in which students work through the same textbook and set of assigned problems, but receive highly interventionist tutorial guidance (Guided) or are free to construct their solutions and obtain feedback only upon demand (Exploratory). We also included a third condition in which students were free to work on problems of their own choosing (Free), to examine the con-

sequences of a completely open-ended discovery environment. The next section describes the three environments in more detail.

Insert Table 1 about here

Guided Environment: Tutored Learning

The guided version of GIL provides tutorial guidance as students learn to solve problems. GIL contains a problem solving model that encodes the target skills in the curriculum and a diagnosis component that compares the student's reasoning to the actions of its own problem solver. The system employs the *model tracing* method (Anderson et al., 1985) and analyzes each step as it is taken to determine whether it is on a path toward a solution or indicates a misconception. GIL stays in the background when the student is following a path leading to a correct solution, but it provides a hint upon a request or an error, enabling the student to continue.

GIL responds to student errors and requests for guidance by accessing the relevant knowledge in the problem solver and generating explanatory feedback. If an error in a student step is found, GIL's explainer analyzes the discrepancies between the student's step and the closest matching correct plan and offers suggestions about how to improve the step. GIL intervenes when students use a programming construct incorrectly, when they incorrectly describe the data that would be input to or output from a particular construct, or when they build a legal portion of a program that is not strategically useful for a solution. GIL first describes what is good about the student's step and then points out the ways in which the step is in error or could be improved by explaining how the student's step deviates from the action considered by the problem solver. Figure 1 displays two levels of help in GIL's reaction to a legal error in the two paragraphs in the Hint Window. The first level provides a hint concerning what is wrong with the step. The student may attempt to fix the error, or may ask for the optional and more directive second level of help, which typically offers a specific suggestion about how to repair the step.

Insert Figure 1 about here

GIL's feedback helps students recover from errors by helping them isolate the errors in a solution and understand why they are incorrect. First, since feedback is provided as soon as students complete each step, students know upon completing each portion of the program whether it is correct and a strategic step toward a solution or must be modified. Second, for steps requiring repair, the feedback indicates whether the step is strategically useful but contains an illegal portion, or is a strategic error and the student should consider a different plan. Third, the feedback focuses the student on the particular components of the step requiring repair.

To emphasize the contrast between the tutoring and discovery learning conditions, we constructed a slightly modified version of GIL for this experiment. This version is designed to tell the student the answer when an error is made, rather than providing a hint, and then following the hint with an answer only upon the student's request. Thus, whenever a student makes an error, the program interrupts and provides not only the first level of help, which offers a hint suggesting what is wrong with the step, but it also automatically generates the second level of help to tell the student how to fix the step. By presenting its own explanation and suggested repair immediately upon errors, this version of GIL differs markedly from a discovery learning environment, in which students learn by forming their own explanations and repairs in response to errors. If there are cognitive benefits to repairing one's own errors, these benefits should be weakest in this condition.

Exploratory Environment: Discovery Learning, Assigned Problems

The second learning environment used in the present study is a version of GIL that provides the reasoning-congruent environment, but without the model tracing constraint and associated tutoring assistance. There are two goals in the design of this environment. First, students working through the same set of problems as the Guided group are given freedom to explore, make mistakes, and diagnose and correct those mistakes themselves. Second, the system is designed to facilitate the process of learning by testing hypotheses.

The exploratory version of GIL provides students more freedom in constructing a solution than the guided version. With model tracing, GIL must check each step for correctness, so the system constrains students to include three components in

each step — a function, the input, and the output. In contrast, the exploratory version of GIL does not require students to complete all three components of one step before beginning another. Students can assemble graph objects (functions and data nodes) in any sequence desired, drawing connections between objects by clicking on one object and dragging a line to another object and releasing the mouse button. Furthermore, the student is not constrained to build graphs only from the bottom up and the top down as they are in the Guided environment. Instead, the student can place any functions and data nodes anywhere in the graph and link them in whatever order desired. Figure 2 displays a partial solution in the exploratory system at a point at which the student decided to request feedback. Note that the student has begun a forward path and has also constructed a portion of the graph that has not yet been connected, using the function *LIST* on the input *a* to get the resulting list *(a)*.

Insert Figure 2 about here

The most important characteristic of the freedom provided in this exploratory version is the freedom to pursue incorrect solutions. In this mode, the system does not interrupt when the student constructs an illegal or non-strategic step. Feedback is available only when specifically requested by the student. For this reason, it is important to provide methods for deleting and editing portions of a graph. In the guided version of GIL, because errors must be corrected as soon as they are made, the only delete capability is the *Oops* button which can be used to cancel the last action or to backup further by successively removing previous steps. The exploratory system contains a *Delete* button that can be used to delete any function or data node or any link between a function and a data node, and a *Replace* button, that can be used to replace one function by another or to enter a new value for a data node.

The second design goal in the exploratory system is to facilitate more explicit hypothesis testing. A crucial component of discovery systems is the ability to elicit feedback from the system when desired, so that students can test hypotheses. The system contains facilities for testing partially constructed solutions and for running a program on given data to examine its behavior. The student can select a *Test*

button and click on any node in the graph. The system then begins with the initial input and tests the paths up to the target node to determine whether each function can be applied to its input data and each data node in the graph is the correct output for the indicated function. If GIL finds any illegally applied functions or incorrect data nodes, it marks the node and prints a message in the Hints window. The Hints Window in Figure 2 contains GIL's response to an error. Since this version of GIL allows the student freedom to explore, it responds only to legal errors when the student requests a test of the graph, and it does not comment on the student's strategy for a solution. Using the test facility, students can construct a partial graph, filling in the data nodes to demonstrate how they expect the program to behave, and can then test the graph to see whether their predictions are correct.

The environment also contains facilities for students to run a complete graph to see whether the sequence of functions behaves as they expect on other examples. The *Run* button causes GIL to test the current graph, and prompts the student to replace the initial input or inputs with new values if it is correct. The system then executes the student's program using the new data, by filling in each data node with the value calculated by operating the function on the new value passed to that function. This enables students to experiment with the behavior of a program by executing the program on new data values. An example of the student running a program is shown in Figure 3.

Insert Figure 3 about here

This version of GIL allows students to explore and work without guidance, but the exploration is within the context of a curriculum of assigned problems. When students have finished a program intended to satisfy the requirements for an assigned problem, they can select the *Submit* button to ask GIL to evaluate their solutions. GIL tests a student's program with a set of examples associated with the problem to determine whether the program computes the correct result for each example. If the student's program satisfies the problem's constraints, then GIL presents the next problem in the curriculum. If the student's program only works for the original example but is not general enough to work for the other examples, GIL provides an example that does not work and asks the student to modify the program.

The hypothesis testing facilities of GIL are enhanced by the use of the graphical representation. Students indicate how the program manipulates its data as they construct a solution. When students test a partially completed solution, they have entered their predicted result for each function application. Thus, students are led to articulate their reasoning and make predictions as they go along. Therefore, the system not only provides facilities for hypothesis testing, but also aids in the articulation of hypotheses, by providing a representation that makes the student's reasoning more explicit (Merrill et al., 1992).

In summary, this environment implements freedom to explore within constrained problems. Students are given the freedom to explore without being interrupted, to make errors, and to determine when to receive feedback, but they are required to solve the same curriculum of problems as the guided students. Students elicit feedback upon demand (as frequently or rarely as desired), rather than being interrupted whenever they make an error. Students must discover and repair their own errors and decide when to request analysis of their solutions. Students in this environment can elicit feedback on the legality of their solution at any time, but can only find out whether their solution fully satisfies the problem specifications when it is complete and submitted.

For the purpose of this experiment, the guided and exploratory versions of GIL have been constructed to heighten the contrast between the tutored and exploratory versions. Thus, the guided version used in this study provides only the model-tracing feedback, but does not offer students the ability to run their own programs or delete or replace arbitrary portions of their graphs. In the complete GIL environment, we have combined the guidance and hypothesis testing facilities so that even when working with model tracing, students can run partially complete solutions to examine the behavior of their programs, and have complete editing control of their solution (Merrill et al., 1992; Reiser et al., 1991).

Free Environment: Discovery Learning, Self-Generated Curriculum

The Free environment provides a facility for students to set their own goals and learning agenda. This implements the third learning condition, discovery learning with a self-generated curriculum. In this version of GIL, students are free to work

on whatever programs they choose. No assignments are displayed, and students are not required to solve problems in a particular curriculum. Thus, their programs are never submitted as solutions. Instead, Free subjects are free to build any program at all, either setting up problems to solve, or just building graphs with no particular problem intended.

In this system, the Submit button is replaced by a *Save* button. Selecting the Save button produces a printed copy of whatever program graph(s) are currently on the screen, and clears that window so that the subject may start anew. Figure 4 displays a student saving the results of some exploration of the LISP functions she had been studying.

Insert Figure 4 about here

The Save feature was added to roughly parallel the notion of completing problems in the first two conditions, in which students are informed when they have completed a problem successfully. We were concerned that Free subjects might be discouraged by the fact that any interesting program graph they might create would have to be simply erased and lost in order to make room for something else. The facility to save and print graphs, which can be used later to study before the posttest, provides a more sensible sequence of interactions than merely building a graph and erasing it. In light of this addition, the Guided and Exploratory versions of GIL were designed to produce printouts of each correct program graph created by subjects. All subjects were told that they would have an opportunity to review their printouts prior to the posttest.

We have designed the Free environment to embody as much as possible the proposed benefits of a total discovery learning environment. The system provides no set goals other than understanding of the material it presents. The student can access all the elements of LISP and discover when each may profitably be used. The student is encouraged to formulate and test hypotheses about LISP and the system contains facilities to help students explicitly articulate their hypotheses and then test them. In addition, students have full control over their interaction with the system and feedback is only provided upon a student's request. Finally, the choice and duration of the learning topics is completely under the student's control, rather than requiring the student to solve problems in a prescribed curriculum.

Method

Subjects learned two chapters of introductory LISP programming, reading the same text but using one of the three learning environments provided guided learning, discovery learning with assigned problems, or self-controlled discovery learning. We evaluated learning consequences by analyzing the records of the students' problem solving, their speed of learning the material, and their performance on a posttest. We assessed motivational consequences via a questionnaire completed at the end of the learning session.

Subjects

The subjects were 30 undergraduates from Princeton University and other nearby colleges and universities (Rider College, Mercer County Community College, and Temple University), recruited through advertisements and paid by the hour for their participation. All had taken no more than one year (or two semesters) of computer programming courses, including no more than one semester course above the high school level and no more than one semester involving a programming language other than BASIC. None of the subjects had any knowledge of LISP. The data from five potential subjects were discarded: three experienced computer malfunctions during the course of the experiment, one left before completing the experiment, and another misunderstood the instructions and failed to produce saved problems. All of the remaining subjects were between 18 and 23 years of age (mean age = 19.3 years). Six subjects were male and twenty-four were female.

Ten subjects were assigned to each of the three experimental conditions (Guided, Exploratory, and Free). In an attempt to minimize differences in performance between conditions that might result from individual differences, we assigned subjects to conditions so as to create a rough balance of programming ability across the conditions. Mathematical reasoning ability is the best available predictor of success in learning to program (Mayer, Dyck, & Vilberg, 1986), so we assigned subjects to conditions such that the distributions of Math SAT scores for each of the three conditions were as similar as possible. The subjects' Math SAT scores ranged from 420 to 800, with a median score of 635. The mean Math SAT scores for each condition were 619, 624, and 612 for the Guided, Exploratory, and Free learning environments respectively.

Apparatus and Materials

The versions of GIL used in this study were implemented in Interlisp and LOOPS and run on Xerox 1108 and 1186 LISP workstations. Subjects completed questionnaires, read written text, and filled out posttests at a desk away from the computers. Interactions with the computers took place in enclosed rooms or cubicles, one subject per area, each containing a single workstation with a standard keyboard, 18" or 20" monochrome display, and an optical three-button mouse for manipulating objects on the screen.

The printed text used to teach subjects LISP is based on the first two chapters of a college-level LISP textbook (Anderson, Corbett, & Reiser, 1987), modified to use GIL's graphical representation for LISP programs. All subjects were given the same instructional text, which consisted of 15 pages of instructional text and diagrams containing approximately 4,600 words.

The assigned problems for the Guided and Exploratory groups included thirteen problems plus one demonstration problem involving manipulation and construction of lists (e.g., see Figure 1). The problems were grouped into three problem sets interspersed within the text. These same problems were also provided to Free subjects as a set of "Optional LISP Exercises." The Free subjects were encouraged to do anything at all which would help them to learn the material in the text. The optional exercises sheet could be used as a source of ideas if desired, and were included to prevent potential discouragement for subjects who might have no idea where to start. These instructions for Free subjects were as follows:

> Keep in mind that you are free to use your time with the computer in any way you please. You are not required to solve any of these problems, nor are you even required to attempt them. You can also work on problems which are extensions or modifications of those presented here, work on problems which are entirely of your own design, or try building program graphs with no particular problem in mind at all. We provide this sheet only because some people like suggestions of possible ideas to get started with.

Procedure

Background Questionnaire: Each subject began by filling out a background ques-

tionnaire containing questions concerning age, SAT scores, and mathematics and computer course experience. The background questionnaire also contained seven questions designed to measure students' attitudes about a variety of learning activities such as learning from lectures, working on mathematics problems, and using computers.

Learning Session: After completing the background questionnaire, each subject was given the first of the three separate sections of printed text to read. Upon reaching the last page, subjects were led through a brief demonstration to familiarize them with the computer-based learning environment. During the demonstration the experimenter led each subject through two solutions of the first problem in the GIL curriculum: one forward reasoning (building a program graph from the input to the output, moving upwards on the screen) and one backward reasoning (building the same graph from the output to the input, moving downwards on the screen). Subjects were instructed that either direction of reasoning could be used whenever desired in a solution. Free subjects were further instructed that it was not necessary to solve any particular problems at all, but that obtaining a specific output from a specific input might be an interesting exercise.

The demonstration also included a run-through of any special buttons of the environment being used (e.g., Oops, Replace, etc.). At the end of the demonstration, all subjects received for future reference a summary sheet outlining the operation of the program.

Following the demonstration, Free subjects were given the remaining two sections of the text and told that, while it might be a good idea to apply new knowledge on the computer before moving on to the next section of text, they were free to use their time with the computer in any way they chose. These subjects were told to learn the material in the text, and to notify the experimenter when they felt comfortable enough with it to be tested. The only stipulation was that the time spent on the computer by each subject had to be at least 90 minutes and no more than 240 minutes, measured from the time at which the subject received the text. These time constraints were based on pilot research with Exploratory subjects, in which most pilot subjects took at least 90 minutes to read the text, receive the demonstration, and complete the thirteen assigned problems, and no subjects took longer than 240 minutes. The time constraints were imposed on Free subjects in order to lessen any differences among conditions which might arise simply as a result

of differences in amount of time spent on learning.

Subjects working with Guided and Exploratory environments were required to solve all the problems in each set. When the subject submitted a correct solution, the computer then presented the next problem. Subjects working with these environments did not receive the second and third sections of printed text until they reached specific points in the problem series. They were free to take as much time as necessary to complete their problem sessions.

Signup Task: After finishing their sessions with the computer, subjects were given the opportunity to sign up for participation in a later experiment using a similar computer system. The experimenter left the subject a signup sheet and told each subject to list their name, phone number, and best time to call if they were interested in participating, or just to leave it blank if not. The experimenter then left the room to pick up printouts and told the subject to leave the form (whether empty or completed) on the desk.

In reality, the signup sheet was intended only to measure subjects' attitudes towards the experience they had just completed. We reasoned that subjects would be more likely to sign up for a second experiment if they were enjoying the first one. Almost all subjects did decide to sign up for the additional experiment, and two of the three who didn't do so cited reasons unrelated to motivation (e.g., one wasn't sure if she'd be able to get a ride again). Therefore, the results of the signup task were excluded from further analysis.

Evaluation Questionnaire: After filling out the signup sheet, each subject completed a more explicit test of motivational orientation in the form of an 11 item "evaluation questionnaire," shown in Table 2. We designed this questionnaire to examine their resulting attitudes toward the domain (including programming itself and the computer learning environment) and toward their own performance, based on three factors that affect motivation in Malone and Lepper's (1987) model — challenge, curiosity, and control.

Insert Table 2 about here

After completing the evaluation questionnaire, subjects were given five minutes to review the printouts of their work on the computer. For Guided and Exploratory

subjects, these printouts depicted the subject's own correct solutions for the assigned problems. For Free subjects, the printouts were those made every time the Save button was pressed.

Posttest: Following the five minute review period, the experimenter removed the printouts and gave the subject the written posttest, designed to measure a variety of LISP programming skills acquired during the learning sessions. Subjects completed all problems on paper, without using any feedback from the computer. Subjects were allowed a maximum of 25 minutes; most subjects finished in somewhat less time. The first problem on the posttest consisted of a complete program graph like those built on the computer, except that the names of all the functions in the graph were missing. Subjects were instructed to fill in the boxes with the names of the functions that would produce the output shown. In the second problem, the function boxes were present and all data nodes were left blank (except for the original input and final output of the program). These two problems assessed whether subjects correctly learned the behavior of the LISP functions described in the text.

The third and fourth problems measured subjects' ability to fix incorrect programs. These problems consisted of complete but buggy program graphs. Subjects were instructed to correct these programs, making a minimum of changes, so that they would work properly.

The last two problems required subjects to create entire program graphs from only an input and an output — these problems were of the same format as those encountered in the learning sessions. However, they required using more complex data structures than the assigned problems for Guided and Exploratory subjects and the optional problem suggestions for Free subjects. These problems required the use of embedded lists, for instance, the list *((a b c) (1 2 3))*. This type of structure was described in the text, and the principles of behavior of the functions described in the text are general enough to handle these structures, but these structures were not used in the exercises in the learning session. Therefore, these problems constitute a near-transfer task.

Results and Discussion

Subjects varied widely both in their performance and in their responses on the motivational questionnaire. Some of this variance is likely to be due to differences

in background and programming aptitude not associated with the differences in the learning environments. Because Math SAT score was our best available measure of students' mathematical reasoning ability, which is the best predictor of individual differences in learning programming (Mayer et al., 1986), we used this score as a covariate in an analysis of covariance to assess differences between conditions.

We performed a second type of analysis to test whether the effects of the instructional manipulations differentially affected students of different ability levels. We assigned each subject an "ability level" on the basis of math SAT score. Those below the median score of 635 (420-630) were classified as the "Low SAT" group and those above the median (640-800) were classified as the "High SAT" group. (We should stress that these terms refer to a measure of programming ability relative to our population of subjects — some "lower ability" subjects were of higher ability than the national average, for example.) We performed an analysis of variance with both SAT level and instructional condition as the independent variables. We also included planned comparisons (described later) to test specific predictions concerning the role of guidance for higher and lower ability students.

The following sections present four sets of analyses. First, we consider the performance of subjects during the learning sessions. Second, we compare subjects' performance on the written posttest. Third, we examine the variety of learning strategies and self-chosen curricula apparent in the behavior of subjects in the two discovery-oriented learning conditions. Finally, we examine the motivational consequences as manifested in subjects' responses to the motivation questionnaire.

Learning Session Performance

Students in the Guided and Exploratory learning conditions were required to solve each problem correctly before receiving the next problem assignment. We defined solution time as the total time to solve the 13 assigned problems following the first demonstration problem and excluding time spent reading the instructional text between problem sets. Mean solution times are displayed in Figure 5. Exploratory subjects took longer to solve the 13 problems than subjects working with the guided version of GIL, $F(1, 17) = 14.58, p = .001$, with mean solution times of 75.8 and 43.5 mins, respectively. This indicates that the tutor's additional guidance helped students solve the problems more quickly than students working with the same

representations without this guidance. Low SAT subjects took longer than High SAT subjects to solve these problems, $F(1, 16) = 23.98, p < .001$, with mean solution times of 82.7 and 36.5 mins, respectively. This large effect supports the use of SAT as a measure of subjects' prior abilities for this domain. The difference between low and high ability subjects was somewhat more pronounced in the Exploratory condition than in the Guided condition $F(1, 16) = 3.15, p = .095$, suggesting that students of lower ability benefitted more from GIL's guidance.

Insert Figure 5 about here

No solution time measure is available for Free subjects because they were not assigned a required set of problems. However, we can measure total learning session times for these subjects, which includes all time spent reading text and working on the computer. The learning session time for Free subjects was 122.1 mins, longer than the Guided group (89.5 mins), but approximately the same duration as the Exploratory group (118.6 mins). More interestingly, the learning session time for Free subjects was negatively correlated with SAT score ($r = -.67$). Because Free subjects were instructed to end their learning sessions when they felt they had learned all of the material, this self-imposed difference in time spent suggests that the subjects may have been accurate in their assessments of how much time they needed to learn.

Next we considered the factors that contributed to the solution time differences between the Guided and Exploratory groups. The longer solution times for Exploratory subjects may indicate that they encountered more difficulty, or alternatively it may indicate that their exploration activities such as running the program to experiment with different inputs simply consumed more time. To address this, we examined the instances in which subjects had to repair their solutions. We counted the occurrences in which subjects received feedback indicating a portion of a solution was in error. This feedback consisted of explanatory error messages for the Guided subjects, and error feedback during a test, run, or submission for the Exploratory subjects. Low ability subjects did receive more error feedback than high ability subjects, 20.2 vs. 6.0 messages per session, respectively, $F(1, 16) = 6.09, p = .03$. However, the amount of error feedback was not reliably different for Guided and Exploratory subjects, 11.5 vs. 14.7 messages, respectively, $F < 1$.

We next considered the cost of repairing errors. If the number of times that subjects received an indication that their solution needed repair did not differ between Guided and Exploratory subjects, then perhaps learning time differences arose because repairs were more difficult for Exploratory subjects. To assess this, we examined the number of functions and data nodes each subject deleted or replaced while constructing and debugging their solutions. Again, low ability subjects had to delete substantially more of their programs than did high ability subjects, 63.1 vs. 16.3 objects, $F(1,16) = 13.04, p = .002$. Furthermore, Exploratory subjects deleted more program components than did Guided subjects, 53.2 vs. 26.2 objects, $F(1,17) = 5.86, p = .03$. The interaction of learning environment with ability level was not reliable, $F(1,16) = 1.45, p = .25$.

These learning session performance measures suggest differences between the two ability groups, and reveal the facilitation in mastering the material provided by model-tracing guidance. First, the low ability subjects encountered substantially more difficulty in mastering the material. They took more than twice as long to solve the assigned problems, encountered three times as many errors, and deleted more than three times as many of their program components while constructing solutions as did the high ability subjects. Second, the results demonstrate the success of the model-tracing guidance. Guided subjects were able to solve problems more easily. This occurred, at least in part, because guided subjects could repair their errors more easily. Presumably the tutor's model-tracing feedback offered during students' solution construction enabled errors to be repaired at a lower cost in deleting and replacing program components than the error feedback obtained by subjects in the Exploratory condition, which required modification of greater portions of their programs. This result is also consistent with Anderson et al. (1993) who found that model-tracing guidance helped students solve problems more quickly than a minimal-feedback group who used the same interface with only minimal feedback on the correctness of completed solutions. The present results extend the findings of this previous study, in that the Exploratory subjects working without model-tracing guidance also had tools specially tailored to testing solutions and discovering errors on their own, and the model tracing still enabled the Guided subjects to solve problems more efficiently.

Posttest Measures

The written posttest was designed to measure students' knowledge of LISP programming acquired during the learning session. The functions, data, and graph generation problems tested coding skills fundamentally important to the creation of program graphs, which was the target skill of the lesson. The functions problem contained eight blank functions to be entered, and the data problem contained five data nodes to be supplied. Each node was scored as either correct or incorrect. The two graph problems were each scored on a five point scale, with one point subtracted for each error or missing step in the complete program. There were no differences between conditions on any of the three coding scores, nor on a combined score derived by averaging them, $F < 1$. The mean proportion correct were .88, .91, and .89 correct for Guided, Exploratory and Free subjects respectively. The high SAT subjects scored slightly better than the low SAT subjects on the combined coding score, .93 versus .86 correct, but the difference was not reliable, $F(1, 24) = 2.35, p = .14$. This difference was not significantly affected by learning environment, $F < 1$. These results suggest that although the Exploratory subjects took longer to solve the assigned problems than Guided subjects, they eventually learned coding skills to the same level of proficiency. The Free subjects also acquired comparable coding skills, even though they were given much greater control over their own learning.

The analysis of debugging scores yielded very different results. The debugging portion of the posttest assesses whether subjects learned to find and repair errors. We computed debugging scores by grading subjects' corrections on the two buggy problems. Each buggy area of a problem graph was worth between 2 and 5 points depending on the complexity of the replacement required to repair the error; there were a total of 23 possible points for the six bugs in the two problems. We assigned corrections made to each buggy area of a program graph a score ranging from zero (representing a failure to notice the bug in the particular location) to the maximum score for that area (for repairs that were functionally correct and sufficiently succinct). A planned comparison revealed that the debugging scores of the Exploratory and Free subjects were significantly higher than those of the Guided subjects $F(1, 26) = 4.31, p = .048$. The proportion correct is shown in Table 3.

Insert Table 3 about here

We also separated the debugging score into two additional measures to separate subjects' abilities to detect errors and to repair them once found. We defined the detection score as the number of the six bugs the subject attempted to repair (whether correctly fixed or not). We defined the bug fixing score as the proportion of points awarded for those bugs that were detected. The two measures are also shown in Table 3. A planned comparison revealed that Exploratory and Free subjects were better at detecting bugs in the posttest programs than were Guided subjects, $F(1, 26) = 5.04, p = .03$. However, subjects in all conditions were approximately equally effective at fixing those bugs which they found, $F < 1$. Performance on these debugging measures did not reliably interact with ability level, $F < 1$. These results suggest that the differences in debugging scores arose primarily from Exploratory and Free subjects' increased ability to detect bugs, and not from any greater proficiency at repairing them. This detection advantage may be a result of the practice discovery learning students receive in finding their errors. While the Guided environment pointed out any errors as they occurred, Exploratory and Free subjects had no such assistance and had to locate bugs on their own.

The locus of the debugging differences in error detection but not error repair is consistent with prior studies of debugging, suggesting that detection and repair skills draw upon separate subskills (Katz & Anderson, 1988; Kessler & Anderson, 1986). These results suggest that once an error is found, repairing it is simply a matter of rewriting the correct portion of the program, requiring the same knowledge used to construct solutions rather than special bug repair knowledge.

Choices of Learning Strategies and Material

Strategies for Selecting Goals

In evaluating the effectiveness of discovery learning in this experiment, it is important to consider how students used the freedom to explore and how their learning progressed without a tutor's guidance. One aspect of freedom in learning is the students' ability to select their own learning strategies. Students learning with no assigned curriculum, and to a lesser extent, those learning with the Exploratory

environment, are free to choose their own strategies for exploring LISP and to determine their own emphases on the various concepts in their exploration. In this section, we consider the strategies evident in the students' learning sessions.

The Free subjects exhibited a wide variety of behaviors ranging from careful solution of the supplied optional problems to undirected exploration. Interestingly, all of the Free subjects attempted at least six of the thirteen problems supplied on the "optional LISP exercises" sheet. The average number of these problems attempted was 10.3. Two subjects attempted all of the supplied problems. Only one subject limited her usage of the computer to the optional problems; the remaining nine subjects engaged in some other form of exploration.

There are several ways in which the students' problem solving in the discovery environments deviated from the more controlled problem solving of the Guided students, even when solving the same problems. First, while some of the Free subjects simply completed the suggested problems and moved on, others appeared to be more actively engaged in the solution process. Some subjects solved a single problem more than once, exploring different ways to produce a correct program (or perhaps simply gaining additional practice which they felt was needed). Some subjects interrupted their solutions of problems to explore issues that arose during the course of a solution. For example, one subject interrupted an attempt to solve one of our problems in order to explore more directly the effects of various functions. After doing so, she returned to the problem she had been working on.

Insert Figure 6 about here

This kind of "intensive self-instruction" was frequently evident in the behavior of the Free subjects. Some subjects built long and sometimes complicated program graphs in order to explore newly learned LISP functions (see Figure 6). These graphs were often much more complex than those produced as solutions to the optional exercises. Other subjects interspersed their work on optional problems to explore the behavior of particular functions without the distraction of a larger problem context (see Figure 4). In this way, subjects were able to provide themselves with focused instruction in important concepts, much as a skilled instructor might interrupt problem solving to focus on poorly understood material.

Concepts Explored

One argument against discovery learning is the concern that students who are left to explore on their own may never discover certain fundamentally important concepts nor practice essential components of the target skill (Ausubel, 1963). Analyses of the Exploratory and Free subject problem solving records provided some support for this concern. The use of concrete examples in the graphical interface of all three learning systems appears to simplify the task of learning how functions behave and how to embed functions to design an algorithm (Reiser et al., 1989). However, the danger of this teaching method is that students may design nongeneral programs — sequences of functions that compute the correct result for the stated example but would not work correctly for other examples. Debriefings with subjects revealed that some understood that their programs would not behave correctly for different input data, but had thought that they would simply change the sequence of functions for different data. This indicates that they did not completely acquire the concept of a program as a fixed sequence of functions that performs a particular calculation on a range of input data.

The three systems respond quite differently to nongeneral programs. In guided mode, GIL's model tracer recognizes any attempts to produce nongeneral solutions after any step, interrupting immediately with meaningful feedback before they can progress. Guided subjects were generally successful in recovering from these errors. This error occurred in a total of six problems in three of the subjects, and was repaired within one or two attempted steps in five of these cases; one occurrence consisted of an extended episode of bad plans of this type. In contrast, the exploratory version of GIL leaves students free to construct nongeneral solutions and does not intervene. When Exploratory students requested feedback, GIL provided feedback only upon illegal statements but did not point out strategic errors. In principle, Exploratory subjects were free to try running their programs on different examples (as in Figure 3) and could detect their nongenerality in that way, but this detection rarely occurred, perhaps because the subjects who constructed such nongeneral solutions did so because they did not understand the manner in which their programs were to be general. Thus, Exploratory subjects obtained feedback on nongeneral solutions predominantly when they submitted them. This feedback proved difficult to understand, and some subjects resubmitted the same nongeneral

solution several times. In the Free environment, of course, no feedback is provided, and so subjects were free to construct nongeneral solutions to suggested problems and may never have realized that these solutions were inappropriate. Indeed, six of the ten Free subjects produced nongeneral solutions to problems from the optional exercises sheet, and of course these six subjects were never informed that there was anything wrong with their solutions. It is possible that these subjects never learned the concept of generality, even though it was mentioned briefly in the text and in the experimenter's demonstration. Unfortunately, the posttest was not constructed specifically to investigate whether students would tend to construct nongeneral solutions, and did not contain problems likely to elicit such solutions.

Some Free subjects also received limited instruction in other areas simply because they chose not to explore them. Just as some of the subjects in this condition reported skipping some of the sample problems because they seemed too easy, other subjects skipped the more complex problems, perhaps because they seemed too difficult and therefore unpleasant. These subjects were thus deprived of extra practice on more challenging assignments.

Despite these examples of Free subjects' failing to discover or practice certain important concepts, it should be noted that they often discovered important concepts that were largely unavailable to subjects in the other conditions. For example, two Free subjects solved some problems by adding on to existing program graphs instead of starting from scratch. The concept of using previously defined functions in building new ones is central to programming but it is not covered in the early part of the GIL curriculum used in this study, so those subjects in the two assigned problems conditions were not able to write programs using their own previously defined programs. In addition, six of the Free subjects and three Exploratory subjects made problem graphs containing embedded lists — a LISP construct that was introduced in the instructional text and examples but not included in any of the required problems (see Figures 4 and 6).

Hypothesis Testing

A crucial issue in evaluating whether subjects made effective use of a discovery learning situation is whether they indeed constructed and tested hypotheses. Subjects in both the Exploratory and Free conditions could construct and test their

hypotheses about LISP, and potentially correct their own specific misconceptions. While it is difficult to identify specific episodes of hypothesis testing in subjects' logs, there is evidence to suggest that Exploratory and Free subjects engaged in this kind of strategy. For example, Free students' in-depth exploration of the properties of specific functions may be an example of hypothesis testing in action.

Explicit evidence for hypothesis testing is provided by students' usage of the Test button, which could be used to determine whether a given portion of a solution graph was correct. On average, Exploratory subjects pressed the Test button approximately once every third step. In some cases, these occurrences were simply checking a completed solution, but in most cases students used the Test button during the construction of a program to evaluate particular hypotheses about the behavior of a partially complete program (as in Figure 2). Many of these tests (35%) uncovered problems in the students' graphs. This suggests that the subjects may have been sensitive to potential difficulties with their solutions. This extensive use of the Test button was not uniform among subjects. Some Exploratory subjects preferred simply to submit their graphs and let the computer find the problems, but most subjects made use of the hypothesis testing capabilities of the system.

The Test button was also used frequently by the Free subjects. Indeed, 57% of the steps taken were followed by tests, and 36% of these tests uncovered illegal statements in the students' programs. Thus, subjects in the two exploratory conditions indeed appeared to use the facilities provided in the system to large extent to evaluate and monitor their own problem solving.

Error Detection

In the previous section on learning consequences, we considered how well subjects exhibited error detection skills on the posttest, and found an advantage for the two discovery learning groups. Here, we consider subjects' performance during the learning sessions, to see how effective they were in analyzing their own errors during the learning session itself.

There are some difficulties in evaluating how well students managed to find their own errors during learning, because there is no clear definition of what steps should be regarded as erroneous. In the Free environment, subjects were encouraged to engage in undirected exploration, so no particular step can truly be classified as an

error without knowing what goal the student had in mind. Even in the Exploratory version of GIL, in which subjects are assigned problems to solve, it is difficult to distinguish between errors and correct steps because a seemingly inappropriate step may lead to a correct yet roundabout solution, or to an exploratory sequence that is not meant to be a correct solution to the assigned problem. It is, however, possible to analyze subjects' own perceptions of errors that need to be fixed. We defined "repair episodes" to be sequences of steps in which a student deleted or replaced a portion of a program graph. Exploratory subjects engaged in an average of 18.5 repair episodes during the learning session. Analyzing the event that occurred immediately prior to each repair episode may reveal the manner in which subjects detected their errors. Table 4 presents a summary of these episodes.

The simplest type of repair episode followed an unsuccessful Submit attempt; 16% of the episodes occurred in this way. In these cases, subjects had either mistakenly believed their programs were correct, or else they simply wanted the computer to uncover any errors in their completed solution. This behavior presumably would not help students acquire the type of error management discovery learning environments are claimed to elicit — students who simply complete a problem and let the computer find their errors for them may have difficulty finding their errors themselves when unassisted. A small number of repair episodes were precipitated by unsuccessful Run attempts. An attempted Run signifies a belief that a program is correct for the data given (GIL only allows students to replace data by a Run in programs that are correct so far), so these episodes also suggest a low level of student involvement with the error detection task.

Insert Table 4 about here

A more interesting group of episodes are those initiated by a Test operation that located an error. This was the most frequent initiating event, comprising 46% of the total episodes. Some of these may have been "lazy tests," made after finishing a problem in order to avoid expending any effort on the location of bugs. Most, however, represented examples of recovering from failed predictions, where students constructed a partial solution which they expected would behave in a particular way, tested it to determine whether it behaved as predicted, and then, upon the

surprising result, were forced to explain the failure and then modified their solution based upon this result. This type of hypothesis-driven learning is precisely the type of behavior discovery learning advocates argue leads to meaningful learning (Bruner, 1961; Papert, 1980; Ranney & Thagard, 1988; Schank & Edelson, 1990). The large proportion of errors corrected in this way suggests that students were making effective use of the hypothesis testing capabilities of the system.

Another interesting group of repair episodes are those that were self-initiated (32% of repair episodes), that is, undertaken without any explicit hint of a problem. In these cases, subjects made errors (or at least believed that they had done so), located their problems, and attempted to repair them. Presumably some of these self-initiated repair episodes were merely corrections of typographical errors or examples of trial-and-error search. Nevertheless, it is interesting that, in a relatively large number of cases, Exploratory subjects determined the need to modify their programs with no assistance, and they could utilize the system's facilities to inform their repair. This behavior, which was not possible for subjects learning with the guided version of GIL, may have contributed to the discovery learning subjects' higher debugging scores on the posttest.

Summary of Learning Consequences

We found no evidence of superior conceptual understanding of LISP functions or a greater ability to construct program graphs for students learning with the discovery environments. The posttest scores provide no evidence to support the claim that concepts learned through discovery will be better understood or more applicable to later situations (Bruner, 1961). In contrast, we found that the Guided group mastered the material more efficiently, requiring less time and fewer edits of their programs than Exploratory subjects. The finding of similar learning outcomes for students receiving different amounts of guidance during learning is consistent with the results of Anderson et al. (1993), who argued that learning depends on the problems students solve correctly and is not strongly affected by the difficulty they encounter in constructing those solutions.

Discovery learning did produce a beneficial effect on the acquisition of debugging skills. Those subjects whose learning took place in more discovery-oriented environments were better at detecting incorrect program graphs. Forcing students to

correct their own errors lengthened the learning process, but it apparently paid off in the form of enhanced error management abilities. It is interesting that superior error management skills appeared to result not only for the Exploratory students, but also for the Free students. The Free students were not required to solve any particular problems, and thus were not constrained to correct any errors in order to proceed through the learning session. These students modified their program graphs only when they did not meet their own goals. Thus, this suggests that subjects in this very free environment indeed set goals for themselves that they attempted to achieve.

These results have demonstrated an advantage for the Exploratory group that has not arisen in previous studies of guidance. Anderson et al. (1993) compared students working with model-tracing guidance to students who received the same feedback only on demand and found faster learning for the guided group but equivalent learning outcomes. Our exploratory environment differs from that used by Anderson et al. in one important respect. The exploratory environment in the present study was constructed as more than simply an equivalent interface without model tracing guidance. The exploratory version of GIL contains support for students to articulate and then test hypotheses. The advantage for the discovery learning groups in debugging may have been facilitated by the tools to support the hypothesis testing phase of debugging, absent in Anderson et al.'s demand-feedback environment.

The exploration allowed by the Free environment appeared to have mixed consequences. Analysis of problem solving interactions showed that increased freedom for exploration resulted in a number of innovative and probably beneficial learning strategies, as subjects constructed their own study sheets to contrast the behavior of functions, constructed more complex combinations of functions than required by the curriculum, explored an additional data construct not required in the assigned curriculum, and solved the same problem in a variety of ways. These are examples of the type of deep, elaborative, integrated processing argued to lead to better learning (Bruner, 1961; Nolen, 1988). The discovery students' use of the Test button for hypothesis testing and subsequent explanation and repair suggests that they may benefit from the freedom to form their own ideas about how programs work (Schank & Edelson, 1990). The finding that subjects often initiated their own repair episodes without outside help shows that they indeed search for their own errors, and that

this process may be helpful to development of error management skills.

In contrast, the important concept of program generality was not addressed by some of the subjects. Some Free subjects constructed programs that were not general and would not have been accepted by GIL in either Guided or Exploratory modes. In addition, some discovery learning subjects constructed awkward and strategically poor programs that would have been prevented in the Guided environment. These findings support the concern that discovery environments can fail to teach students important information simply because the students do not stumble upon it (Ausubel, 1963; Kurland & Pea, 1985). However, it does not appear that the coding portion of the learning posttest was sensitive enough to detect any potential decrements in programming ability of the discovery learning students.

Taken together the results demonstrate both the potential advantages of discovery learning for providing opportunities for students to manage their own learning and acquire error management skills, the cost in learning time and effort, and the potential danger of discovery learning for those students who do not take advantage of the opportunity to explore, and who therefore may learn inferior strategies or even fail to acquire important concepts.

Motivational Consequences

We measured subjects' attitudes toward the domain and themselves with the eleven item questionnaire shown in Table 2. We consider these results in light of the factors proposed by Malone and Lepper (1987) in their discussion of the characteristics of learning situations that elicit intrinsic motivation. The eleven items are grouped into five subcomponents reflecting attitude toward the domain of LISP, attitude toward the student's own performance, perceived difficulty, perceived amount of freedom, and attitude toward the computer's feedback.

The central question addressed in these analyses concerns the motivational effects of providing guidance or allowing exploration in a learning environment designed to cover a target curriculum. Therefore, the major focus in this analysis is to examine the motivational consequences of guidance or freedom to explore for students who are solving a common set of problems. Thus, we focus our analyses primarily on the Guided and Exploratory groups. Following this set of results, we will also present a comparison of the Free subjects with the other two groups to consider the potential

motivational effects of leaving students free to construct their own curriculum.

We performed an analysis of variance with both SAT level and learning environment as the independent variables. Some studies of aptitude-treatment interactions have suggested that lower ability students perform better with more structure in a curriculum, while higher ability students exhibit a truncated or even opposite effect of constraint (Snow & Lohman, 1984). Therefore, rather than look for overall effects of instructional condition, we examined the effects of guidance versus exploration separately for low and high ability students. To do this, we performed three planned comparisons to examine (1) the effects of learning environment for low ability students, (2) the effects of learning environment for high ability students, and (3) the hypothesis that the environment would have opposite effects for the two ability groups.

Intrinsically motivating tasks are those students find enjoyable and are interested in pursuing for their own sake rather than for external rewards. The first two questions of our questionnaire were intended as general assessments of students' resulting attitudes toward the domain. The questions asked subjects to rate their enjoyment of the learning experience and their interest in learning more about programming. The results, shown in Figure 7, suggest that the learning environments differentially affected high and low ability students, although this effect was not reliable, $F(1, 24) = 1.68, p = .21$. There is the suggestion that the Exploratory environment led to a more negative attitude toward the domain than the Guided environment for low ability students, $F(1, 24) = 2.25, p = .15$. The learning environment had no reliable effect on high ability students' attitudes toward the domain, $F < 1$. Although the pattern of differential outcomes for guidance versus exploration for the high and low ability students is not strongly supported by this analysis, the pattern is consistent with the other measures we will present in this section.

Insert Figure 7 about here

The third, fourth, and fifth questions asked subjects to rate their abilities and their overall performance in the learning task. These questions asked subjects to assess their performance in the task relative to their expectations, their performance relative to others, and their ability to learn similar material in the future, relative

to others (see Figure 8). Malone and Lepper (1987) suggested that an optimal level of challenge elicits intrinsic motivation because it increases students' self-esteem. On this measure, the guidance or freedom differentially affected the attitudes of low and high ability students toward their performance, $F(1,24) = 4.94, p = .04$. The separate contrasts for low and high ability students suggest that guidance produced more positive outcomes for low ability students, $F(1,24) = 2.71, p = .11$, while there is a suggestion of a negative effect of guidance for high ability students, but this difference was not reliable, $F(1,24) = 2.24, p = .15$.

Insert Figure 8 about here

It seems the freedom or guidance in the environment led to two different outcomes in assessing competence for the two ability groups. Unlike the Guided system, which assisted students with their problem solving as much as they required, the Exploratory system afforded more opportunities for students to produce satisfying solutions on their own, or to experience difficulties in correctly solving the problems. The interaction contrast indicates that the two ability groups responded differentially to these opportunities. One possible interpretation is that the Exploratory system's lack of intervention gave higher ability students the opportunity to demonstrate their competence to themselves, but low ability students were led to feel less competent when they experienced difficulty in constructing acceptable solutions.

The next component of the motivation ratings, consisting of the sixth, seventh, and eighth questions, asked subjects to assess how well they met the perceived difficulty of the learning task. Subjects were asked to rate the difficulty of writing their programs, how well they understood each problem, and how easily they were able to accomplish their solutions. In Malone and Lepper's analysis, an optimal level of challenge and clearly-defined and appropriately difficult goals are important components of an intrinsically motivating task. The ratings are displayed in Figure 9. Here again, the guidance and freedom had quite different effects on the the low and high ability students' perceptions of the task difficulty, $F(1,24) = 10.71, p = .003$. The low ability students in the Exploratory condition found the task significantly more difficult than those in the Guided environment, $F(1,24) = 8.11, p = .009$. In marked contrast, however, the high ability subjects in the Exploratory condition tended to find the task *easier* than their peers in the Guided environment,

$F(1, 24) = 3.17, p = .09$. The interaction in these judgments, like the student's attitude toward their performance, may result from the opportunities the Exploratory environment offers for students to assess their own abilities and success. High ability subjects may have found the Exploratory environment easier than the Guided environment because they more frequently experienced evidence of their ability to cope successfully with the problem solving task without help. These subjects apparently were able to take advantage of the Exploratory environment's added flexibility in order to set and achieve their own goals. In contrast, the low ability students found themselves struggling more often in the Exploratory environment, and felt less able to handle the complexity of the learning task. Thus, the tutor's guidance greatly helped low ability subjects to structure their learning experience and cope with the new task's difficulty, but offered fewer opportunities for high ability students to prove themselves capable of tackling the domain's challenges.

Insert Figure 9 about here

The next two subcomponents of the questionnaire assessed subjects' attitudes toward two aspects of the environment that Lepper and Malone suggest are important factors in intrinsically motivating tasks. The ninth question asked subjects how free they were to solve problems in the way they wanted; it was intended to measure perceived control over the teaching interaction. The mean ratings, shown in Figure 10, exhibit a pattern similar to the previous measures, but the differences are not reliable, all Fs < 1. It is interesting that students' feelings of freedom were not affected by the presence or absence of model-tracing feedback while subjects were working on the assigned problems. Although these two environments differed drastically in the amount of freedom they provided, the students did not feel appreciably more free in the Exploratory learning system. It is possible that these subjects, with no prior experience with interactive learning environments, had little basis on which to evaluate the freedom provided.

Insert Figure 10 about here

The final subcomponent, consisting of the tenth and eleventh questions, assessed the amount and helpfulness of performance feedback, another factor in Malone and Lepper's (1987) analysis of intrinsic motivation. The two contributing questions assess the perceived amount of feedback (from "not enough" through "just right" up to "too much") and the helpfulness of that feedback (from "not helpful" to "very helpful"). The mean ratings are displayed in Figure 11. The pattern suggests that the guidance and freedom differentially affected the low and high ability subjects' attitudes toward feedback, $F(1,24) = 2.50, p = .13$. In fact, low ability students perceived the feedback to be less frequent and helpful in the Exploratory environment, $F(1,24) = 4.99, p = .04$, while high ability subjects exhibited no differences in their attitudes toward the computer's feedback in the two environments, $F < 1$. Here again, low ability subjects exhibited more dissatisfaction with their ability to meet the challenge of the task's difficulty in the Exploratory environment.

Insert Figure 11 about here

Finally, we consider the attitudes of the Free subjects. So far, we have seen that the low ability subjects generally exhibit more negative attitudes in the Exploratory environment than the Guided environment, while the high ability subjects either do not show a difference or else exhibit more positive attitudes in the Exploratory environment. The mean ratings collapsed across all five subcomponents of the motivation questionnaire are shown in Figure 12. Interestingly, the results of the Free subjects do not parallel the results of the Exploratory subjects. Any negative effect on attitudes of exploration in the low ability subjects appears greatly truncated in the Free condition, and there is no evidence of any positive effect of freedom to explore in the Free condition for the high ability subjects. The lack of required problems appears to have ameliorated both the potential positive and negative outcomes of the freedom to explore. During the learning session, the Free subjects received no external feedback yet on how they were performing, since they did not submit their solutions to be checked, nor had they yet taken the posttest to assess their understanding of the material.

Insert Figure 12 about here

Summary of Motivational Consequences

We did not find overwhelming support for increased development of intrinsic motivation for students learning by discovery. Instead, the nature of the result depended upon the relative ability of the student. To summarize these results, we first focus on the contrast between the two groups who were required to solve a set of assigned problems, the Guided and Exploratory groups.

Our results suggest that discovery learning creates more opportunities for attributions concerning the student's performance. The general pattern of results was a differential effect of learning environment, with the suggestion of lower ratings for low ability students in the Exploratory environment than the Guided environment, and no such effect or a trend toward higher ratings in the Exploratory environment for the high ability students.

We suggest that these results can be explained by considering the opportunities for self-attribution each environment provides. The critical attributions concern overcoming obstacles. Discovery learning creates more opportunities to attribute success or failure in overcoming errors. Whereas the Guided system provided frequent feedback, the Exploratory environment enabled students to detect and attempt to overcome obstacles on their own. The stakes are higher, and so are the attributional consequences. This creates the potential for both positive and negative consequences. We found strong evidence for differential effects for low and high ability students. First, consider the low ability students, who tended toward more negative attitudes in the Exploratory environment than in the Guided environment. Those in the Exploratory situation tended to be less satisfied with their ability to master the domain (as measured by perceived difficulty), less satisfied with the computer's assistance, and tended (not reliably however) toward more negative opinions of the domain. Although all of these low ability students correctly solved the problems, the learning task was indeed objectively more difficult in the Exploratory condition, as evidenced by the time required to solve the problems (almost twice as long as the Guided students), the number of errors encountered, and the cost of repairing each error. The Guided students were able to recover from errors more easily, because the tutor would suggest an error fix in many cases, or at least would tell students whether their attempted fixes were correct. In addition, the Guided students were able to resolve planning impasses more easily — when Guided stu-

dents didn't understand what to do next, they could always try a step and get the tutor's reaction. In contrast, the Exploratory students would have to follow a plan far enough along to see its consequences and then decide for themselves whether it seemed to be leading to a solution. While these challenges created events that could potentially lead to positive attributions for students who could overcome these difficulties, they seemed to lead to negative outcomes for the low ability students for whom the difficulties were more taxing. Presumably the discovery learning condition led to more cases of students feeling they had made mistakes and feeling confused about how to proceed. Therefore, these students were led to feel less able and less motivated to engage further in the domain.

Interestingly, although the Guided students received a good deal of assistance, these interventions did not lead to a negative impact on low ability students' self-esteem. One might expect that students who realize they are receiving a great deal of help would attribute their success to the tutor's assistance rather than their own abilities. Indeed, for the high ability students, noting they received even a small amount of help negatively affected their self-attributions, leading to ratings of the learning in the Guided condition as more difficult than when they were left free to explore. However, this did not seem to occur for the low ability students. Indeed, the low SAT students' attitudes toward the domain and their performance tended to be more positive in the Guided than in the Exploratory environments, in some cases equal to or perhaps greater than the attitudes of the high ability subjects.

This suggests that tutoring may serve as a buffer against the difficulty of the learning task for low ability students. Although the tutor did indeed point out many cases in which students made errors (on average 17 errors in the 13 problems for the low SAT students), the quick intervention limited the consequences of errors, and may have helped students avoid frustrating episodes in recovering from difficult errors. Furthermore, the knowledge that the tutor will immediately notify students upon errors or initiation of bad plans provides feedback students can use to monitor their problem solving and may help students feel more confident as they construct a solution — any step accepted by the tutor is guaranteed to be part of a potential solution. This may have contributed to the reduction of stress as students tried to construct solutions. At the end of the learning session, when students are asked to consider how difficult the task was and how well they performed, they may undervalue the tutor's contribution to their success, and therefore attribute their

success to their mastery over the domain. Thus, low ability Guided students rate their performance more highly than comparable students using the Exploratory environment.

By intervening and limiting the consequences of errors, GIL exhibits some advantages of expert human tutors, who offer constant (but subtle) feedback to help guide students' problem solving (Merrill, Reiser, Ranney, & Trafton, 1992; Merrill, Reiser, Merrill, & Landes, 1993). GIL is not designed to be as "gentle" as human tutors in their interventions. Indeed in this experiment, we made GIL even more directive in its feedback by automatically providing both a hint and the suggested correction in the Guided mode rather than letting students receive the suggested fix only if they so request. Even so, the guidance in constructing solutions and limiting consequences of errors appears to have helped students feel more in control of their learning and generally more positive about themselves and the domain, just as human tutors support students' learning (Lepper, Aspinwall, Mumme, & Chabay, 1990; Merrill et al., 1992, 1993).

For high ability students, this challenge provided by the Exploratory environment appeared not have the same negative consequences. The high ability students rated their success more highly in the Exploratory environment (as evidenced by perceived difficulty), and they did not exhibit the lowered performance judgments or dissatisfaction with the computer's assistance that low ability subjects experienced in the Exploratory condition. Presumably these high ability students were better able to overcome the obstacles when encountering errors in the Exploratory environment, and interpreted the experience as indicating successful application of their abilities. Apparently being able to meet the challenge of the Exploratory environment led to greater feelings of competence then being interrupted by the tutor to be told how to repair errors. Even though these students could have fixed the errors by themselves without frustration (as evidenced by the positive attributions of the high ability Exploratory subjects), these Guided students were less satisfied with their performance. Relying on a tutor's help may lead students to assume that they needed this help, just as it may lead observers to attribute lower abilities to students receiving a teacher's unsolicited assistance (Graham & Barker, 1990). Hence, although the Exploratory condition was objectively more difficult in that it took longer to solve the assigned problems and students encountered more errors, in fact it was perceived as less difficult by the high ability students. This result

is consistent with the interpretation that the Exploratory environment led to more intrinsic motivation. Intrinsically motivated students show greater persistence in the face of failure (Dweck & Leggett, 1988). Although the Exploratory learning situation contained more difficulties, these obstacles were less negatively interpreted than those encountered by comparable students in the tutored condition.

Our experiment does not directly examine why the high and low ability students responded differently to the freedom in the learning environment. One possible explanation for this is that the important differences between students concern differences in ability, and the low SAT group performed more negative attributions in the Exploratory learning situation because they encountered more failures, while the high ability students found themselves better able to handle the slight increased cost in errors and time. The suggestion of an interaction in the solution time results, with larger effects of guidance for the lower ability students, suggests that although even the high ability students did encounter more errors in the Exploratory than the Guided environment, these errors were more easily handled than were the errors encountered by the low ability students. Another possibility is that the attitudes the students bring to the learning task affects the attributions they make. Perhaps characteristics of the high ability students such as higher self-esteem or more focus on learning goals made them more resilient to failure and obstacles. Either or perhaps both of these explanations may underlie the differential effects of freedom and guidance for the two ability groups. What is important for the current study is that these two ability groups differently responded to the freedom.

The findings of positive and negative influences on students' assessments of their competence and ability is a very important result. Attributing success and failure to ability rather than to effort influences students' expectations of future success (Weiner, 1979, 1985). Furthermore, students' judgments of their abilities affect future effort and the manner in which they will react to obstacles. Students who attribute failures to their lack of ability show reduced effort and less resistance to obstacles in future performance (Bandura, 1982; Dweck & Leggett, 1988).

The Free learning condition, in which students construct their own learning agenda, produced a different pattern of events. Unlike the Guided and Exploratory groups, these students were not required to solve a set of problems, and thus the learning session does not contain the same sequence of clear successes or failures that must be explained. The effect of this situation appeared to weaken both the positive

motivational effects of discovery for the high ability students and the negative outcomes for the low ability students. The challenges met by the high ability students were only those that were self-generated, and these did not appear to be as rewarding as being able to solve the assigned problems. Thus, this version of discovery learning did not lead to the same motivational advantages over the Guided condition found in the Exploratory environment. Similarly, for the low ability students, the challenges were not as intimidating as for the assigned problems condition. If a problem appeared too difficult, students could chose not to undertake its solution. Thus, the free curriculum discovery learning situation did not lead to the negative motivational outcomes found in the assigned problems environment.

Finally, an interesting issue concerns the students' assessments of freedom. Being free to set goals, make mistakes and repair them, and find one's own solutions are argued to be important components of discovery learning. We found no evidence that students in this experiment were sensitive to the relative amount of freedom provided. Indeed, although the differences were not reliable, the pattern roughly mirrored the pattern of ratings for performance and problem difficulty. Perhaps students' assessments of freedom depend upon the freedom to be successful.

The Interplay of Cognitive and Motivational Factors

The desirability of structuring learning so that students engage in a task because it is rewarding rather than for external rewards has long been a theme in pedagogical debates. Recent theoretical frameworks have provided a context to investigate the consequences of these types of motivations. Intrinsic motivation has been a topic of modern educational concern since an important set of studies in the 1970's began to document the potential negative effects of reward, demonstrating that rewards extrinsic to the learning task may undermine students' subsequent intrinsic interest in the task (e.g., Condry & Chambers, 1978; Deci & Ryan, 1985; Lepper & Greene, 1978). A related dimension concerns the focus of students on learning goals, learning for its own sake, versus performance goals, performing to receive positive feedback or avoid negative feedback from others (Dweck, 1986; Dweck & Leggett, 1988). Lepper (1988) reviewed several types of influences that increased intrinsic motivation and learning goal focus could have on learning outcomes. Students intrinsically motivated commit more time to engaging in a task (Lepper & Greene, 1978), are more

likely to choose challenging tasks (Condry & Chambers, 1978; Pittman et al., 1982), and show greater persistence in the face of failure (Dweck, 1975; Dweck & Leggett, 1988). Each of these influences on students' strategies or investment of effort may plausibly result in more effective learning. Recently Lepper and his colleagues have attempted to demonstrate a direct link between increasing the intrinsic motivation of a learning environment and the learning gains of the students. In several studies Lepper and his colleagues demonstrated that increasing the intrinsic motivation of an educational game by setting the challenges in a fantasy context (drawing geometric shapes versus an astronaut searching for new planets in space) led to increased learning gains about the domain (Logo and geometry) even when time on task was controlled (Lepper & Cordova, 1992; Parker & Lepper, 1992). In some of these studies, there was also a tendency for the motivationally embellished games to lead to somewhat more positive attitudes toward the domain and their competence.

Our results are consistent with and extend this prior work on the potential attitudinal and learning outcomes of different learning environments. First, the studies by Lepper and his colleagues manipulated intrinsic motivation by setting the game activity within situation contexts such as pirates searching for buried treasure or astronauts looking for planets. While these contexts are clearly "fantasy contexts" in that the children must use their imaginations to view the animation on the screen as representing pirates or astronauts and treasure or planets, these contexts may also serve to better link the new mathematical problem solving activities concerning geometric shapes to their prior intuitions than the decontextualized problem solving task of drawing abstract shapes in Logo. Thus, the task may not only have been more motivating but may also have served to better ground the new skills on students' prior conceptions (c.f., Confrey, 1990; Lampert, 1986). We adopted a different strategy, manipulating the potential intrinsic interest in the activity by directly manipulating the opportunities for challenge and control of the problem solving task, while keeping the situation context constant (all problems involved manipulations of simple data such as numbers and lists of words).

The results of the present study suggest the subtle nature of the motivational effects of the learning environment design. The greater freedom in the Exploratory environment clearly provided more potential for challenge and student control. However, this did not uniformly result in increased intrinsic motivation and positive attitudes for all students. Instead, the increased challenge and freedom appeared to

lead to more negative attitudes for lower ability students, and comparable or more positive attitudes for the high ability students. We have proposed that the increased freedom provides more opportunities for students to resolve obstacles themselves because the guidance helps identify potential errors and prevents them from becoming serious. Higher ability students appear to meet these challenges more successfully, while they appear to cause differentially more difficulty and negative attitudinal differences for the low ability students. These results have demonstrated the potential for influencing students' motivational outcomes through the design of features of the task argued to be central to intrinsic motivation, namely challenge and control (Malone & Lepper, 1987). However, the results also suggest that designing an optimal level of challenge and control needs to be sensitive to the backgrounds and abilities of individual students. These results demonstrate the importance of considering the fit between an environment and students' strategies and abilities (Shute, 1992, 1993). Shute (1992, 1993) has demonstrated that learning outcomes in an interactive environment depend in part on the fit of students' strategies to the amount of control allowed by the environment.

In the present study, rather than keeping time on task constant as in Lepper's studies (Lepper & Cordova, 1992; Parker & Lepper, 1992), we included an assigned curriculum for the Guided and Exploratory groups, which may be more representative of typical instructional situations, in which students must work through a specified set of material. The required material may have heightened the potential for negative consequences, forcing students to encounter obstacles and to overcome them in some manner before continuing.

The present experiment also reveals the importance of disentangling the cognitive and motivational effects. While the direction of the motivational consequences appeared to differ depending on the ability level of the student, we found no such differential effects for cognitive outcomes. We found solely positive effects of guidance on the problem solving difficulty measures. The guidance appeared to help subjects solve problems with less difficulty than the more free exploratory system. Earlier demonstrations of learning efficiency advantages for model-tracing guidance (Anderson et al., 1993) found an advantage for a guided group over a minimal-feedback group who used the same interface with only minimal feedback on the correctness of solutions. The present experiment suggests that providing tools specifically tailored to scaffold exploration, such as the environment's support for making predictions

explicit in the notation and for testing predictions (Merrill & Reiser, 1993; Merrill et al., 1992), may result in more positive attitudes than a guided environment, for students inclined and able to successfully handle these opportunities, despite the somewhat less efficient learning sessions. Thus, the advantages of guidance, found in this and other studies (Anderson et al., 1993), must be weighed against the potential positive advantages of providing support and freedom to explore.

Conclusions

We began with a discussion of the arguments for discovery learning, and some reasons for concern about these claims. By constructing variations of a computerized interactive learning environment, we can begin to provide empirical evidence to investigate both the cognitive and motivational consequences of discovery and guided learning in a fashion heretofore impossible. Our results have provided direct evidence that students' judgments of their own abilities and their resulting interest in a domain will be affected by the type of support offered or freedom allowed by a learning environment.

Our results suggest a view of discovery learning as a situation that creates important opportunities for learning outcomes. What occurs with these opportunities appears to depend upon the abilities and perhaps upon the attitudes and expectations students bring to the learning situation. Some students take advantage of the freedom in the discovery learning situation to initiate effective learning strategies that may enable them to investigate more sophisticated concepts than the guided students, or to investigate the concepts in a depth not possible in the guided learning situation. In addition, these students manage their own learning and learn to recognize their own errors, thus acquiring better error management skills than the guided learning students. The freedom to fail enables positive attributions for those students who succeed. The high ability students tended to form more positive attributions about their performance than students of similar ability in the guided learning situation, which provided direction in overcoming obstacles they may have been able to resolve on their own.

With the freedom of discovery learning also comes opportunities for negative outcomes. Some students failed to employ these types of effective strategies, and indeed seemed to fail to construct examples sufficient to acquire one of the more

important constructs in the curriculum. Furthermore, the increased opportunities for attributions about one's ability to overcome obstacles led to more negative outcomes for the low ability students. These students struggled to construct solutions and faced with a large amount of evidence of their own difficulties, formed more negative opinions both about the domain and about their own abilities in the domain. For these students, the increased opportunities to demonstrate their effectiveness led to more negative inferences.

In contrast to the discovery learning situation, the guided learning appeared to insulate the students from assessing their own abilities, by providing fewer opportunities for students to overcome obstacles on their own, while also preventing the consequences of errors or impasses from becoming too severe. For low ability students, the computer tutor appears to have some advantages similar to the scaffolding provided by human tutors, in that the frequent interventions enable students to continue solving difficult problems they otherwise could not handle. Tutored students feel like they have been the ones responsible for the performance success, rather than attributing their success to the tutor, with corresponding negative attributions about their own abilities (Lepper et al., 1990).

There can be no clear answer as to whether discovery learning or guided learning is "superior." Certainly the answer must depend on the goals of the learning and the particular domain. The present experiment also suggests the important contribution of the student's own ability and attitudes. Discovery learning provides more opportunities for students to control their own learning. The potential cognitive benefits of this control are the opportunity to employ more effective learning strategies and to better learn error management skills, but the potential costs are the acquisition of less efficient strategies and the possibility of failing to exercise important components of the skill. The potential motivational benefits are increased confidence in one's ability to handle challenges and perhaps an increased interest in the domain, but the potential costs are conclusions about one's ineffectiveness and a corresponding loss of interest in the domain. Overall, with more at stake, there appears to be more to gain for the high ability students, but also more to lose for the low ability students. The present results strongly indicate the importance of making guidance available in a learning environment for students who may need or want it.

Acknowledgements

We are grateful to Douglas Merrill for discussions concerning this research and for assistance with the statistical analyses. We are also grateful to Marcia Johnson and Margaret Recker for comments on a previous draft of this manuscript, and to Mark R. Lepper for helpful suggestions concerning the motivational measures used in this research. This research was supported in part by contracts MDA903-87-K-0652 and MDA903-90-C-0123 from the Army Research Institute to Princeton University, contract MDA 903-92-C-0114 from the Army Research Institute to Northwestern University, and a research grant from the James S. McDonnell Foundation to Princeton University. This research was performed using equipment donated by the Xerox Corporation University Grant Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of these agencies and institutions.

References

Anderson, J. R. (1983). *The architecture of cognition.* Harvard University Press, Cambridge, MA.

Anderson, J. R. (1987). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review, 94*, 192–210.

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science, 228*, 456–462.

Anderson, J. R., Conrad, F. G., & Corbett, A. T. (1993). The LISP Tutor and skill acquisition. In Anderson, J. R. (Ed.), *Rules of the mind*, pp. 143–164. Erlbaum, Hillsdale, NJ.

Anderson, J. R., & Corbett, A. T. (1993). Tutoring of cognitive skill. In Anderson, J. R. (Ed.), *Rules of the mind*, pp. 235–255. Erlbaum, Hillsdale, NJ.

Anderson, J. R., Corbett, A. T., & Reiser, B. J. (1987). *Essential LISP.* Addison-Wesley, Reading, MA.

Ausubel, D. P. (1963). *The psychology of meaningful verbal learning: An introduction to school learning.* Grune and Stratton, New York.

Bandura, A. (1982). Self-efficacy mechanism in human agency. *American Psychologist, 37*, 122–147.

Bereiter, C., & Scardamalia, M. (1989). Intentional learning as a goal of instruction. In Resnick, L. B. (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glaser*, pp. 361–392. Erlbaum, Hillsdale, NJ.

Brown, A. L., Bransford, J. D., Ferrara, R. A., & Campione, J. C. (1983). Learning, remembering, and understanding. In Flavell, J. H., & Markman, E. M. (Eds.), *Mussen's handbook of child psychology, Volume 3*, pp. 77–166. Wiley, New York.

Bruner, J. S. (1961). The act of discovery. *Harvard Educational Review, 31*, 21–32.

Bruner, J. S. (1966). *Toward a theory of instruction.* Belknap Press/Harvard University Press, Cambridge, MA.

Burton, R. R., & Brown, J. S. (1982). An investigation of computer coaching for informal learning activities. In Sleeman, D. H., & Brown, J. S. (Eds.), *Intelligent tutoring systems*, pp. 79–98. Academic Press, London.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine learning*, Vol. 2. Morgan Kaufmann, Los Altos, CA.

Carroll, J. M., Mack, R., Lewis, C., Grischkowsky, N., & Robertson, S. (1985). Exploring exploring a word processor. *Human-Computer Interaction, 1*, 283–307.

Carver, S. M. (1988). Learning and transfer of debugging skills: Applying task analysis to curriculum design and assessment. In Mayer, R. E. (Ed.), *Teaching and learning computer programming.* Erlbaum, Hillsdale, NJ.

Charney, D. H., & Reder, L. M. (1986). Designing interactive tutorials for computer users. *Human-Computer Interaction, 2*, 297–317.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science, 13*, 145–182.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In Mandl, H., & Lesgold, A. (Eds.), *Learning issues for intelligent tutoring systems*, pp. 1–18. Springer-Verlag, New York.

Collins, A., & Stevens, A. L. (1982). Goals and strategies of inquiry teachers. In Glaser, R. (Ed.), *Advances in instructional psychology, Volume 2*, pp. 65–119. Erlbaum, Hillsdale, NJ.

Condry, J., & Chambers, J. (1978). Intrinsic motivation and the process of learning. In Lepper, M. R., & Greene, D. (Eds.), *The hidden costs of reward*, pp. 61–84. Erlbaum, Hillsdale, NJ.

Confrey, J. (1990). A review of the research on student conceptions in mathematics, science, and programming. In Cazden, C. B. (Ed.), *Review of research in education*, Vol. 16, pp. 3–56. American Educational Research Association, Washington, D.C.

Cronbach, L. J. (1966). The logic of experiments on discovery. In Shulman, L. S., & Keislar, E. R. (Eds.), *Learning by discovery: A critical appraisal.* Rand McNally and Company, Chicago, IL.

Dalbey, J., & Linn, M. C. (1985). The demands and requirements of computer programming: A literature review. *Journal of Educational Computing Research, 1*, 253–274.

Davis, R. B. (1966). Discovery in the teaching of mathematics. In Shulman, L. S., & Keislar, E. R. (Eds.), *Learning by discovery: A critical appraisal*, pp. 114–128. Rand McNally and Company, Chicago, IL.

Deci, E. L., & Ryan, R. M. (1985). *Intrinsic motivation and self-determination in human behavior.* Plenum Press, New York, NY.

Dugdale, S. (1982). Green Globs: A microcomputer application for graphing of equations. *Mathematics Teacher, 75*, 208–214.

Dweck, C. S. (1975). The role of expectations and attributions in the alleviation of learned helplessness. *Journal of Personality and Social Psychology, 31*, 674–685.

Dweck, C. S. (1986). Motivational processes affecting learning. *American Psychologist, 41*, 1040–1048.

Dweck, C. S., & Leggett, E. L. (1988). A social-cognitive approach to motivation and personality. *Psychological Review, 95*, 256–273.

Elsom-Cook, M. (1990). Guided discovery tutoring. In Elsom-Cook, M. (Ed.), *Guided discovery tutoring: A framework for ICAI research*, pp. 3–23. Paul Chapman Publishing, London.

Graham, S., & Barker, G. P. (1990). The down side of help: An attributional-developmental analysis of helping behavior as a low-ability cue. *Journal of Educational Psychology, 82*, 7–14.

Katz, I. R., & Anderson, J. R. (1987-1988). Debugging: An analysis of bug location strategies. *Human-Computer Interaction, 3*, 351–399.

Kessler, C. M., & Anderson, J. R. (1986). A model of novice debugging in LISP. In Soloway, E., & Iyengar, S. (Eds.), *Empirical studies of programmers*, pp. 198–212. Ablex, Norwood, NJ.

Klahr, D., Dunbar, K., & Fay, A. L. (1990). Designing good experiments to test bad hypotheses. In Shrager, J., & Langley, P. (Eds.), *Computational models of scientific discovery and theory formation*, pp. 355–402. Morgan Kaufmann Publishers, Inc., Palo Alto, CA.

Klayman, J., & Ha, Y.-W. (1987). Confirmation, disconfirmation, and information in hypothesis testing. *Psychological Review, 94*, 211–228.

Kurland, D. M., Clement, C. A., Mawby, R., & Pea, R. D. (1987). Mapping the cognitive demands of learning to program. In Pea, R. D., & Sheingold, K. (Eds.), *Mirrors of minds: Patterns of experience in educational computing.* Ablex, Norwood, NJ.

Kurland, D. M., & Pea, R. D. (1985). Children's mental models for recursive LOGO programs. *Journal of Educational Computing Research, 1*, 235–244.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence, 33*, 1–64.

Lampert, M. (1986). Knowing, doing, and teaching multiplication. *Cognition and Instruction, 3*(4), 305–342.

Lepper, M. R. (1988). Motivational considerations in the study of instruction. *Cognition and Instruction, 5*, 289–309.

Lepper, M. R., Aspinwall, L., Mumme, D., & Chabay, R. W. (1990). Self-perception and social perception processes in tutoring: Subtle social control strategies of expert tutors. In Olson, J. M., & Zanna, M. P. (Eds.), *Self inference processes: The sixth Ontario symposium in social psychology*, pp. 217–237. Erlbaum, Hillsdale, NJ.

Lepper, M. R., & Chabay, R. W. (1985). Intrinsic motivation and instruction: Conflicting views on the role of motivational processes in computer-based education. *Educational Psychologist, 20*, 217–230.

Lepper, M. R., & Cordova, D. I. (1992). A desire to be taught: Instructional consequences of intrinsic motivation. *Motivation and Emotion, 16*, 187–208.

Lepper, M. R., & Greene, D. (1978). Overjustification research and beyond: Toward a means-ends analysis of intrinsic and extrinsic motivation. In Lepper, M. R., & Greene, D. (Eds.), *The hidden costs of reward*, pp. 109–148. Erlbaum, Hillsdale, NJ.

Lepper, M. R., & Malone, T. W. (1987). Intrinsic motivation and instructional effectiveness in computer-based education. In Snow, R. E., & Farr, M. J. (Eds.), *Aptitude, learning, and instruction: Volume III Conative and affective process analyses.* Erlbaum, Hillsdale, NJ.

Lesgold, A., Lajoie, S., Bunzo, M., & Eggan, G. (1992). SHERLOCK: A coached practice environment for an electronics troubleshooting job. In Larkin, J. H., & Chabay, R. W. (Eds.), *Computer assisted instruction and intelligent tutoring*

systems: Shared goals and complementary approaches, pp. 201 – 238. Erlbaum, Hillsdale, NJ.

Lewis, M. W., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology, 17*, 26–65.

Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. In Snow, R. E., & Farr, M. J. (Eds.), *Aptitude, learning, and instruction: Volume III Conative and affective process analyses.* Erlbaum, Hillsdale, NJ.

Mayer, R. E. (1985). Learning in complex domains: A cognitive analysis of computer programming. In Bower, G. H. (Ed.), *The psychology of learning and motivation*, Vol. 19, pp. 89–130. Academic Press, New York.

Mayer, R. E., Dyck, J. L., & Vilberg, W. (1986). Learning to program and learning to think: What's the connection?. *Communications of the ACM, 29*, 605–610.

Merrill, D. C., & Reiser, B. J. (1993). Scaffolding learning by doing with reasoning-congruent learning environments. In *Proceedings of the Workshop in Graphical Representations, Reasoning and Communication from the World Conference on Artificial Intelligence in Education (AI-ED '93)*, pp. 9–16 Edinburgh, Scotland. The University of Edinburgh.

Merrill, D. C., Reiser, B. J., Beekelaar, R., & Hamid, A. (1992). Making processes visible: Scaffolding learning with reasoning-congruent representations. In Frasson, C., Gauthier, G., & McCalla, G. I. (Eds.), *Intelligent Tutoring Systems: Second International Conference, ITS '92*, pp. 103–110 New York. Springer-Verlag.

Merrill, D. C., Reiser, B. J., Merrill, S. K., & Landes, S. (1993). Tutoring: Guided learning by doing. Tech. rep. No. 45, The Institute for the Learning Sciences, Northwestern University, Evanston, IL.

Merrill, D. C., Reiser, B. J., Ranney, M., & Trafton, J. G. (1992). Effective tutoring techniques: A comparison of human tutors and intelligent tutoring systems. *The Journal of the Learning Sciences, 2*, 277–306.

Nisbett, R. E., & Ross, L. (1980). *Human inference: strategies and shortcomings of social judgment.* Prentice-Hall, Englewood Cliffs, NJ.

Nolen, S. B. (1988). Reasons for studying: Motivational orientations and study strategies. *Cognition and Instruction, 5*, 269–287.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc., New York.

Parker, L. E., & Lepper, M. R. (1992). Effects of fantasy contexts on children's learning and motivation: Making learning more fun. *Journal of Personality and Social Psychology, 62,* 625–633.

Pea, R. D., Kurland, D. M., & Hawkins, J. (1987). Logo and the development of thinking skills. In Pea, R. D., & Sheingold, K. (Eds.), *Mirrors of minds: Patterns of experience in educational computing.* Ablex, Norwood, NJ.

Pittman, T. S., Emery, J., & Boggiano, A. K. (1982). Intrinsic and extrinsic motivational orientations: Reward-induced changes in preference for complexity. *Journal of Personality and Social Psychology, 42,* 789–797.

Ranney, M., & Reiser, B. J. (1989). Reasoning and explanation in an intelligent tutoring system for programming. In Salvendy, G., & Smith, M. J. (Eds.), *Designing and using human-computer interfaces and knowledge based systems: Proceedings of the Third International Conference on Human-Computer Interaction,* pp. 88–95. Elsevier Science Publishers, New York.

Ranney, M., & Thagard, P. R. (1988). Explanatory coherence and belief revision in naive physics. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society,* pp. 426–432 Montreal.

Reiser, B. J., Beekelaar, R., Tyle, A., & Merrill, D. C. (1991). GIL: Scaffolding learning to program with reasoning-congruent representations. In *The International Conference of the Learning Sciences: Proceedings of the 1991 conference,* pp. 382–388 Evanston, IL. Association for the Advancement of Computing in Education.

Reiser, B. J., Friedmann, P., Gevins, J., Kimberg, D. Y., Ranney, M., & Romero, A. (1988). A graphical programming language interface for an intelligent LISP tutor. In *Proceedings of CHI'88 Conference on Human Factors in Computing Systems,* pp. 39–44 New York.

Reiser, B. J., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (1992). Knowledge representation and explanation in GIL, an intelligent tutor for programming. In Larkin, J. H., & Chabay, R. W. (Eds.), *Computer-assisted instruction and intelligent tutoring systems: Shared goals and complementary approaches,* pp. 111–149. Erlbaum, Hillsdale, NJ.

Reiser, B. J., Ranney, M., Lovett, M. C., & Kimberg, D. Y. (1989). Facilitating students' reasoning with causal explanations and visual representations. In Bierman, D., Breuker, J., & Sandberg, J. (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence and Education*, pp. 228–235 Springfield, VA. IOS.

Resnick, L. B., & Ford, W. W. (1981). *The psychology of mathematics for instruction.* Erlbaum, Hillsdale, NJ.

Resnick, M., & Ocko, S. (1991). LEGO/Logo: Learning through and about design. In Harel, I., & Papert, S. (Eds.), *Constructionism*, pp. 141–150. Ablex, Norwood, NJ.

Roschelle, J. (1992). Learning by collaboration: Convergent conceptual change. *The Journal of the Learning Sciences*, *2*, 235–276.

Scardamalia, M., Bereiter, C., McLean, R. S., Swallow, J., & Woodruff, E. (1989). Computer-supported intentional learning environments. *Journal of Educational Computing Research*, *5*, 51–68.

Schank, R. C. (1986). *Explanation patterns: Understanding mechanically and creatively.* Erlbaum, Hillsdale, NJ.

Schank, R. C. (1990). Case-based teaching: Four experiences in educational software design. *Interactive Learning Environments*, *1*, 231–253.

Schank, R. C., & Edelson, D. J. (1989/1990). A role for AI in education: Using technology to reshape education. *Journal of Artificial Intelligence in Education*, *1*, 3–20.

Schank, R. C., & Jona, M. Y. (1991). Empowering the student: New perspectives on the design of teaching systems. *The Journal of the Learning Sciences*, *1*, 7–35.

Schank, R. C., & Leake, D. B. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence*, *40*, 353–385.

Schauble, L., Glaser, R., Raghavan, K., & Reiner, M. (1991). Causal models and experimentation strategies in scientific reasoning. *The Journal of the Learning Sciences*, *1*, 201–238.

Schoenfeld, A. H. (1988). Mathematics, technology, and higher order thinking. In Nickerson, R. S., & Zodhiates, P. P. (Eds.), *Technology in education: Looking toward 2020*, pp. 67–96. Erlbaum, Hillsdale, NJ.

Schwartz, J. L. (1989). Intellectual mirrors: A step in the direction of making schools knowledge places. *Harvard Educational Review, 59*, 51–61.

Schwartz, J. L., & Yerushalmy, M. (1987). The geometric supposer: Using microcomputers to restore invention to the learning of mathematics. In Perkins, D. N., Lochhead, J., & Bishop, J. (Eds.), *Thinking: The second international conference on cognition and categorization*, pp. 525–536. Erlbaum, Hillsdale, NJ.

Shute, V. J. (1992). Aptitude-treatment interactions and cognitive skill diagnosis. In Regian, J. W., & Shute, V. J. (Eds.), *Cognitive approaches to automated instruction*, pp. 15–43. Lawrence Erlbaum Associates, Hillsdale, NJ.

Shute, V. J. (1993). A comparison of learning environments: All that glitters . . .. In Lajoie, S. P., & Derry, S. J. (Eds.), *Computers as cognitive tools*, pp. 47–74. Lawrence Erlbaum Associates, Hillsdale, NJ.

Shute, V. J., & Glaser, R. (1990). A large-scale evaluation of an intelligent discovery world: Smithtown. *Interactive Learning Environments, 1*, 51–77.

Shute, V. J., Glaser, R., & Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In Ackerman, P. L., Sternberg, R. J., & Glaser, R. (Eds.), *Learning and individual differences*, pp. 279–326. W. H. Freeman and Company, New York.

Snow, R. E., & Lohman, D. F. (1984). Toward a theory of cognitive aptitude for learning from instruction. *Journal of Educational Psychology, 76*, 347–376.

Suchman, J. R. (1961). Inquiry training: Building skills for autonomous discovery. *Merrill-Palmer Quarterly, 7*, 147–169.

Swan, K. (1989). Logo programming and the teaching and learning of problem solving. *Journal of Artificial Intelligence in Education, 1*, 73–92.

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science, 12*, 257–285.

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*, 58–89.

Trafton, J. G., & Reiser, B. J. (1991). Providing natural representations to facilitate novices' understanding in a new domain: Forward and backward reasoning in programming. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, pp. 923–927 Chicago, IL.

Trafton, J. G., & Reiser, B. J. (1993). The contributions of studying examples and solving problems to skill acquisition. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pp. 1017–1022 Boulder, CO.

VanLehn, K. (1990). *Mind bugs: The origins of procedural misconceptions.* MIT Press, Cambridge, MA.

VanLehn, K., Jones, R., & Chi, M. T. H. (1992). A model of the self-explanation effect. *The Journal of the Learning Sciences, 2*, 1–59.

Weiner, B. (1979). A theory of motivation for some classroom experiences. *Journal of Educational Psychology, 71*, 3–25.

Weiner, B. (1985). An attributional theory of achievement motivation and emotion. *Psychological Review, 92*, 548–573.

White, B. Y. (1993). Thinkertools: Causal models, conceptual change, and science education. *Cognition and Instruction, 10*, 1–100.

Wittrock, M. C. (1966). The learning by discovery hypothesis. In Shulman, L. S., & Keislar, E. R. (Eds.), *Learning by discovery: A critical appraisal.* Rand McNally and Company, Chicago, IL.

Yerushalmy, M., Chazan, D., & Gordon, M. (1990). Mathematical problem posing: Implications for facilitating student inquiry in classrooms. *Instructional Science, 19*, 219–245.

Table 1: Learning Environments Used in the Study

- *Guided Environment: Tutored Learning.* GIL monitors students' problem solving to interrupt whenever students make errors and provide feedback designed to help students understand and fix the error.

- *Exploratory Environment: Discovery Learning, Assigned Problems.* This version of GIL contains the same problem solving interface, but it does not interrupt upon errors to offer guidance as the tutored version of the system does. Instead, it contains facilities to enable students to test their ideas.

- *Free Environment: Discovery Learning, Self-Generated Curriculum.* The same exploratory learning version of the system is used, but there are no assigned problems. Students are free to explore by building whatever programs they desire or solving any problems of their own choice.

Table 2: Motivation Questionaire

1. How would you rate your overall experience learning LISP today?
   (1 – "unpleasant," 4 – "neutral," 7 – "lots of fun")
2. Are you interested in learning more about computer programming?
   (1 – "no, not at all", 4 – "maybe", 7 – "yes, very interested")
3. How do you feel about your performance on the learning task you have just completed?
   (1 – "much worse than I would have expected," 4 – "about as well as I would have expected," 7 – "much better than I would have expected")
4. How well do you think you did relative to others learning programming by this method?
   (1 – "far below average," 4 – "about average," 7 – "far above average")
5. How would you rate your ability to learn computer programming in the future compared to other students with similar background?
   (1 – "far below average," 4 – "about average," 7 – "far above average")
6. How hard was it to write programs on the computer today?
   (1 – "too hard," 4 – "just right," 7 – "too easy")
7. How well did you understand each problem and how you were expected to solve it?
   (1 – "not at all, very unclear," 7 – "very well, no problem there")
8. To what extent were you able to accomplish the steps you wanted to accomplish while using the computer?
   (1 – "very difficult to do so," 7 – "very easy to do so")
9. How free were you to solve problems in the way you wanted?
   (1 – "not free, solutions controlled by computer," 7 – "very free to come up with own solutions")
10. Please rate the amount of feedback and intervention provided by the computer:
    (1 – "not enough, it should have said more," 4 – "just right," 7 – "too much, it should have left me alone more")
11. When the computer did provide instructional feedback, how helpful was it?
    (1 – "not helpful," 4 – "somewhat helpful," 7 – "very helpful")

Table 3: Performance on the Debugging Posttest

	Guided	Exploratory	Free
Overall Debugging	.67	.83	.80
Bug Location	.78	.92	.92
Bug Repair	.84	.92	.86

Table 4: Exploratory Subjects' Repair Episodes

Self-initiated:	59	(32%)
Initiated by successful Run:	2	(1%)
Initiated by unsuccessful Test:	86	(46%)
Initiated by unsuccessful Run:	8	(4%)
Initiated by unsuccessful Submit:	30	(16%)
Total:	185	

Figure Captions

Figure 1. GIL's explanatory feedback in response to an illegal step.

Figure 2. Testing a partial program in the exploratory version of GIL

Figure 3. Exploring the behavior of a program by running it on new data

Figure 4. Saving the result of exploration in the Free version of GIL

Figure 5. Time to correctly solve the assigned problems.

Figure 6. Complex exploration in the Free learning environment.

Figure 7. Rated attitude toward the domain (enjoyment of the lesson and interest in learning more).

Figure 8. Subjects' attitudes toward their own performance, assessed by asking subjects how well they performed relative to their expectations and relative to other students.

Figure 9. Subjects' judgments of how well they met the task difficulty, assessed by asking subjects how difficult they found the programs to write, how well they understood the problems, and how well they were able to accomplish their solutions.

Figure 10. Ratings of the amount of freedom available during the learning session.

Figure 11. Ratings of the amount of feedback and helpfulness of the feedback provided by the computer. High ratings on the two contributing questions represent perceptions of frequent and helpful feedback, respectively.

Figure 12. Mean ratings across all five subcomponents of the attitude measures for all three learning environments. High ratings represent positive attitudes.

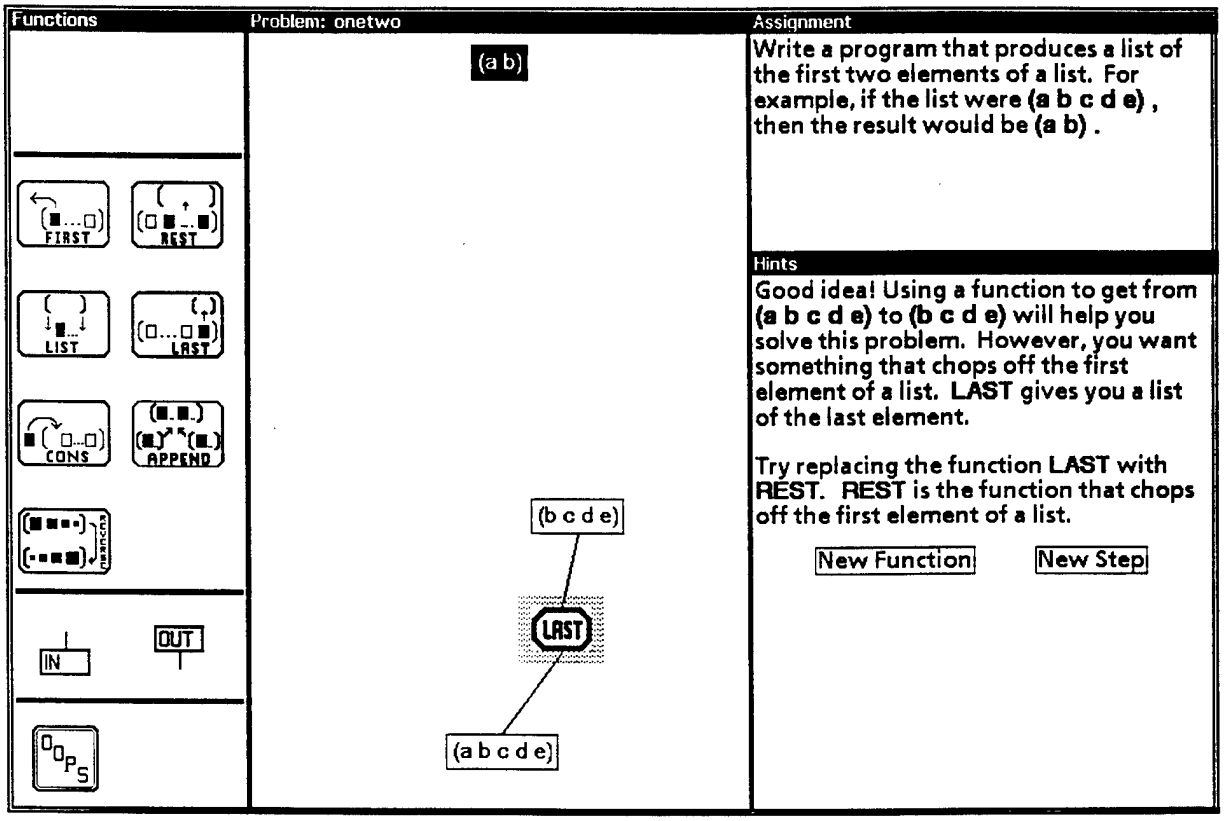
Functions
Problem: onetwo
Assignment
(a b)
Write a program that produces a list of the first two elements of a list. For example, if the list were (a b c d e), then the result would be (a b).
FIRST
REST
LIST
LAST
CONS
APPEND
IN
OUT
Hints
Good idea! Using a function to get from (a b c d e) to (b c d e) will help you solve this problem. However, you want something that chops off the first element of a list. LAST gives you a list of the last element.
Try replacing the function LAST with REST. REST is the function that chops off the first element of a list.
New Function
New Step
(b c d e)
LAST
(a b c d e)

Fig 1

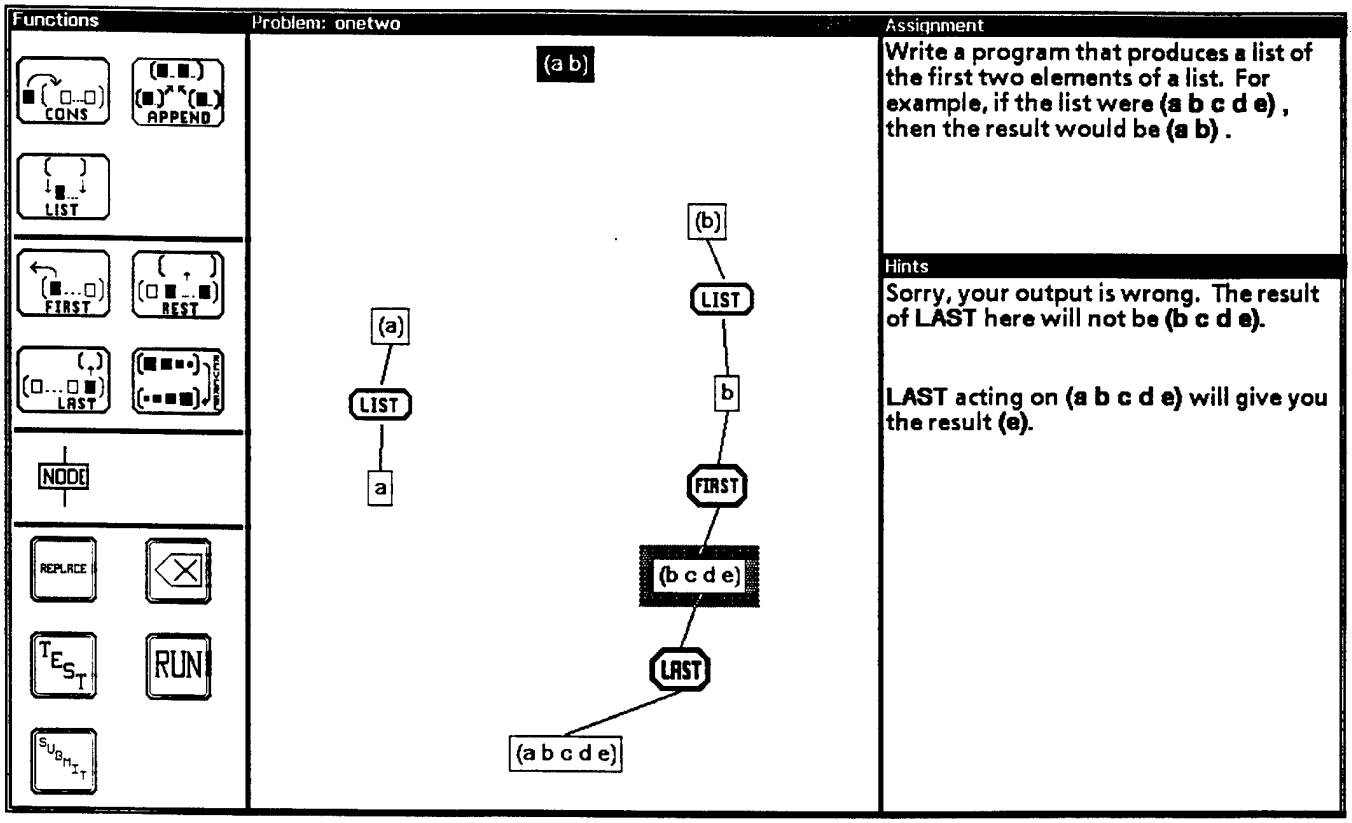
Functions
Problem: onetwo
Assignment
CONS
APPEND
LIST
FIRST
REST
LAST
NODE
REPLACE
RUN
(a b)
(a)
LIST
a
(b)
LIST
b
FIRST
(b c d e)
LAST
(a b c d e)
Write a program that produces a list of the first two elements of a list. For example, if the list were (a b c d e), then the result would be (a b).
Hints
Sorry, your output is wrong. The result of LAST here will not be (b c d e).
LAST acting on (a b c d e) will give you the result (e).

FIG 2

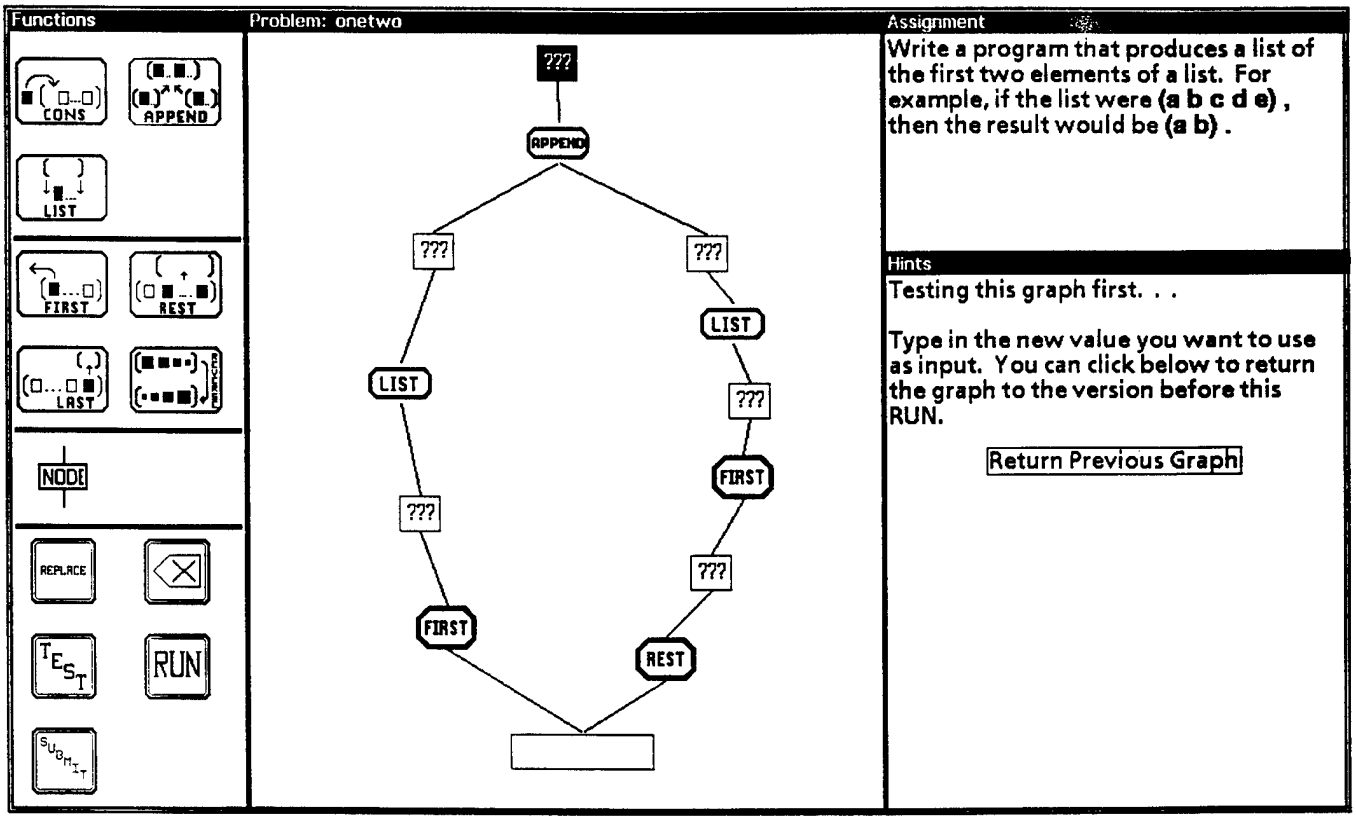
Functions
Problem: onetwo
Assignment
Write a program that produces a list of the first two elements of a list. For example, if the list were (a b c d e), then the result would be (a b).
Hints
Testing this graph first. . .
Type in the new value you want to use as input. You can click below to return the graph to the version before this RUN.
Return Previous Graph
CONS
APPEND
LIST
FIRST
REST
LAST
NODE
REPLACE
RUN
???
APPEND
LIST
FIRST
REST

FIG 3

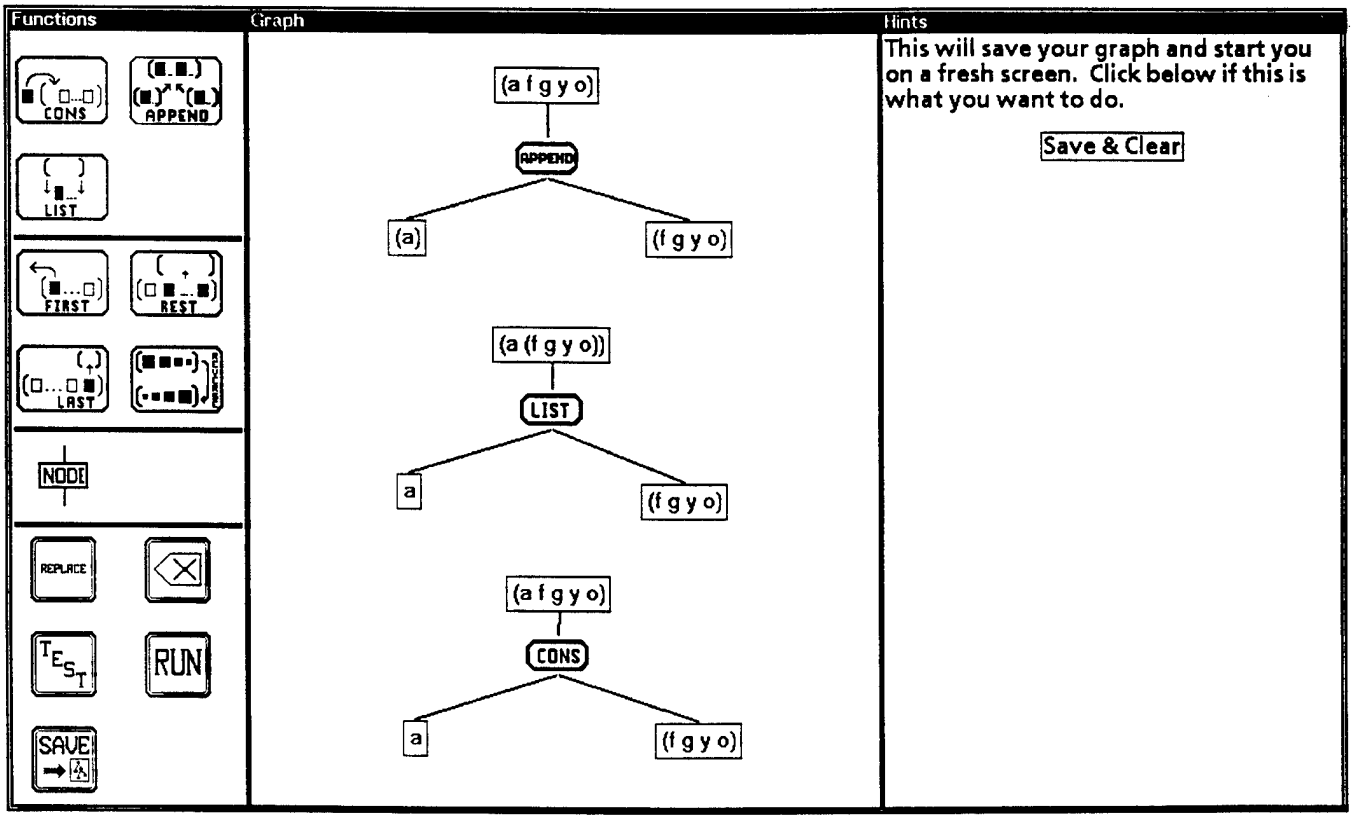
Functions
Graph
Hints
CONS
APPEND
LIST
FIRST
REST
LAST
NODE
REPLACE
TEST
RUN
SAVE
(a f g y o)
APPEND
(a)
(f g y o)
(a (f g y o))
LIST
a
(f g y o)
(a f g y o)
CONS
a
(f g y o)
This will save your graph and start you on a fresh screen. Click below if this is what you want to do.
Save & Clear

Fig 4

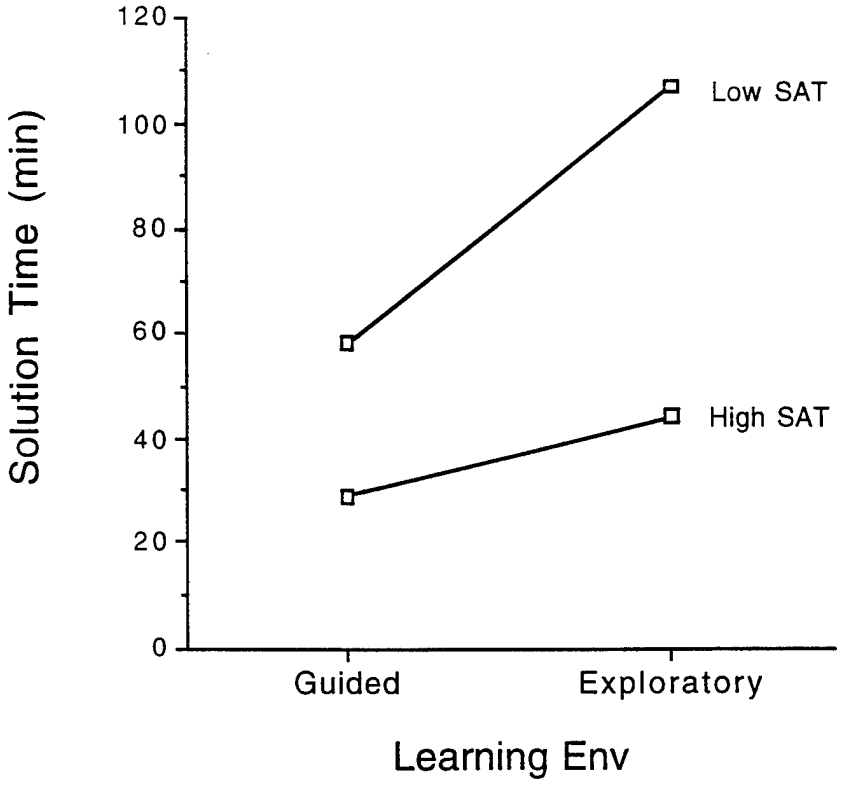
120
100
80
60
40
20
0
Solution Time (min)
Low SAT
High SAT
Guided
Exploratory
Learning Env

FIG 5

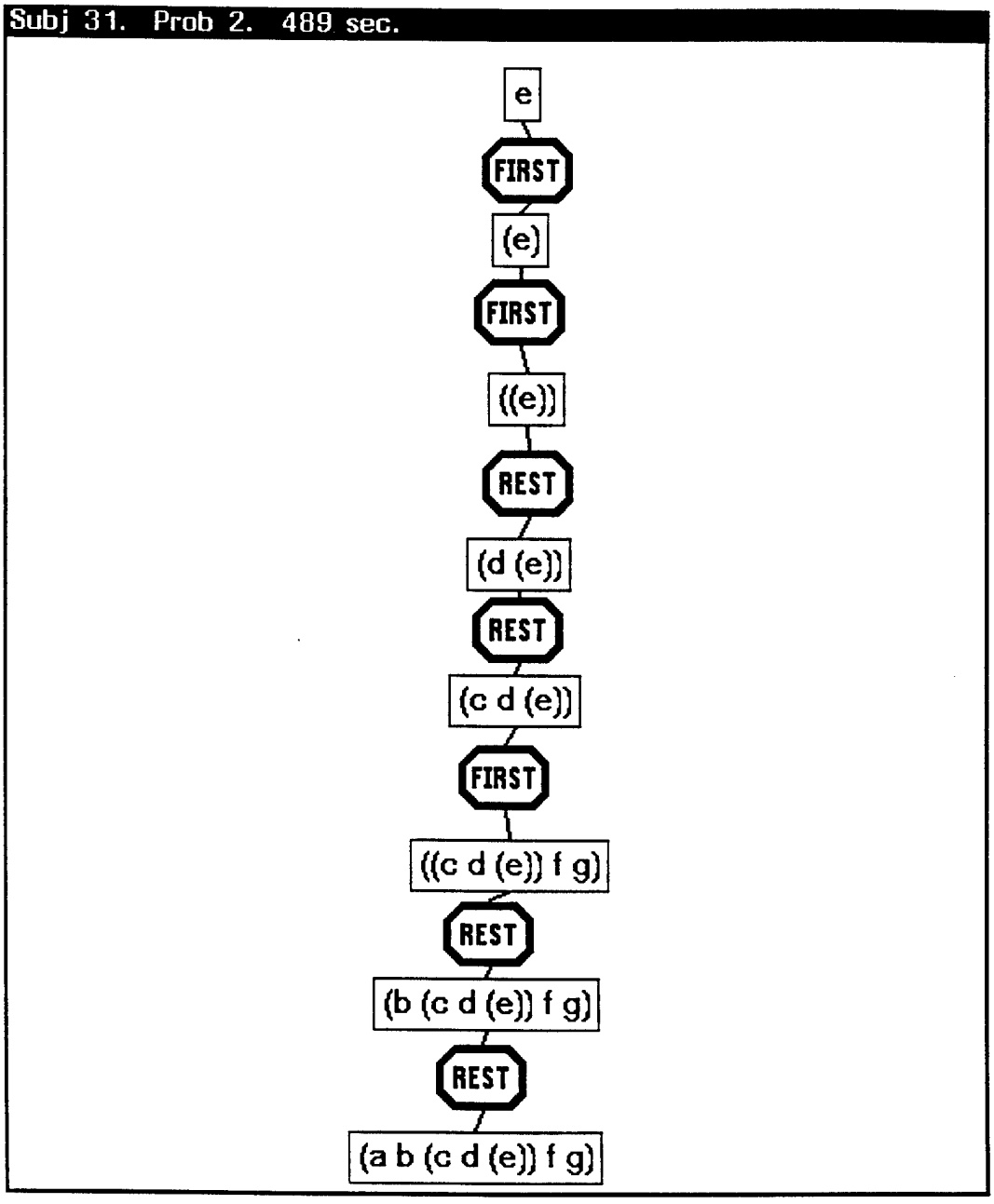
Subj 31. Prob 2. 489 sec.
e
FIRST
(e)
FIRST
((e))
REST
(d (e))
REST
(c d (e))
FIRST
((c d (e)) f g)
REST
(b (c d (e)) f g)
REST
(a b (c d (e)) f g)

Fig 6

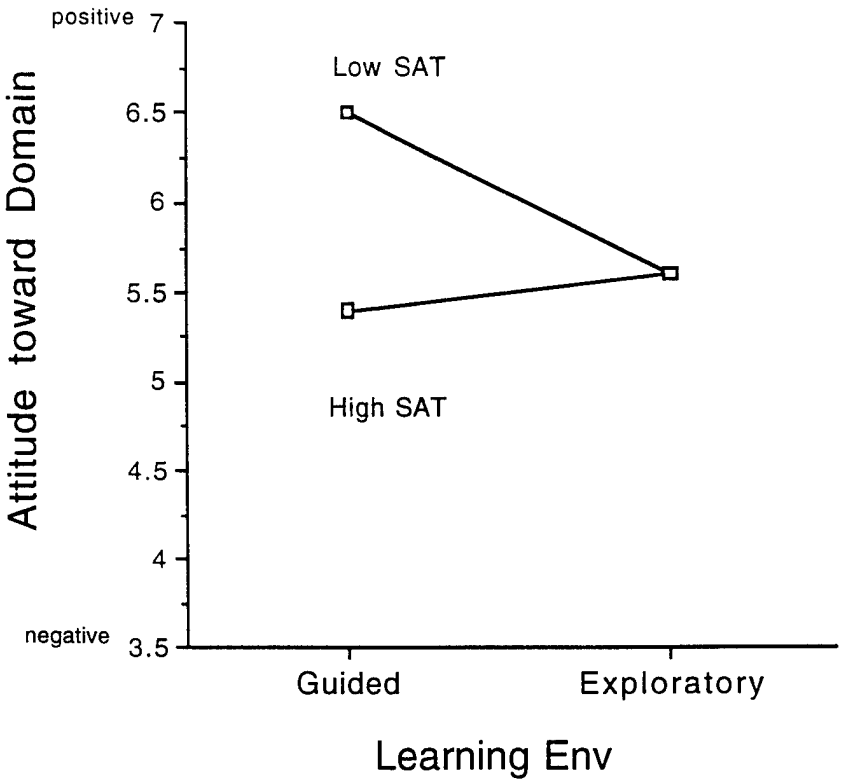
positive
7
6.5
6
5.5
5
4.5
4
negative
3.5
Attitude toward Domain
Low SAT
High SAT
Guided
Exploratory
Learning Env

Fig 7

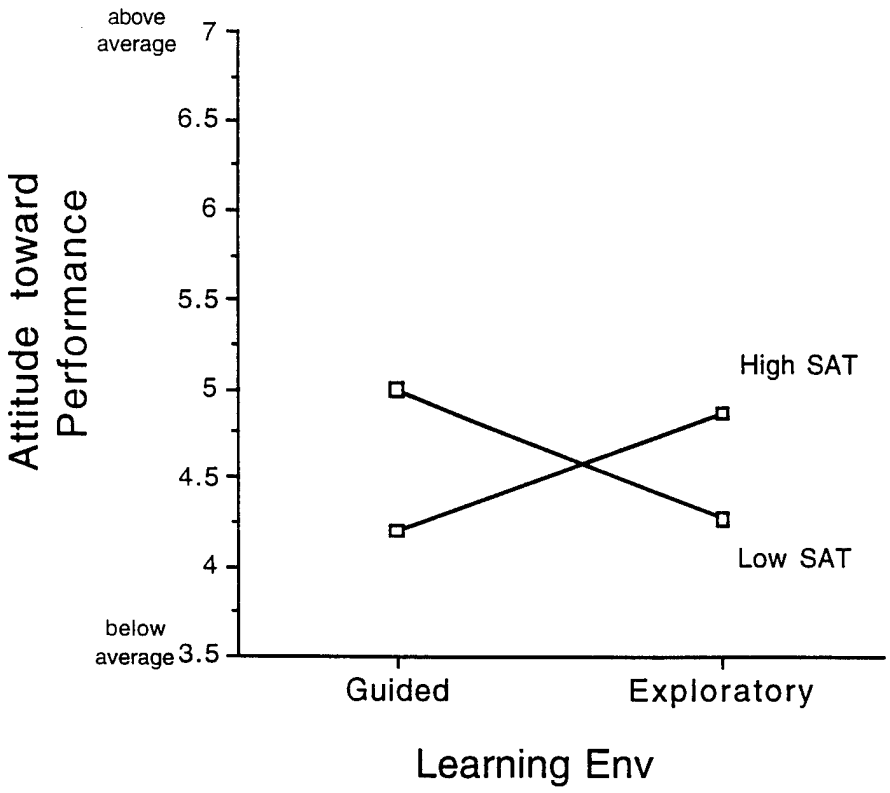
above average
7
6.5
6
5.5
5
4.5
4
below average
3.5
Attitude toward Performance
High SAT
Low SAT
Guided
Exploratory
Learning Env

FIG 8

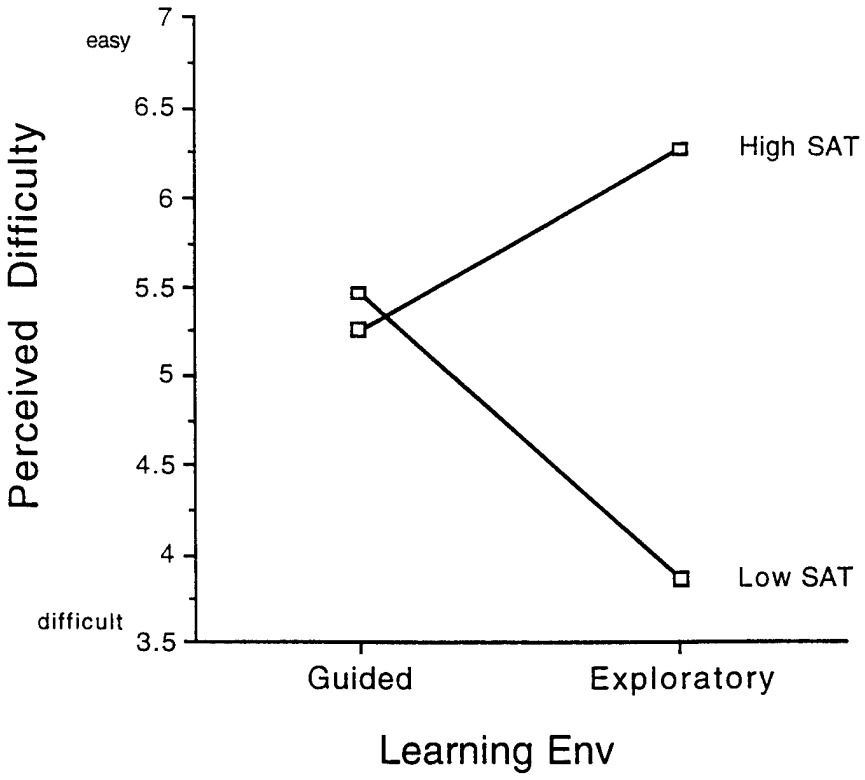
Perceived Difficulty
easy
difficult
7
6.5
6
5.5
5
4.5
4
3.5
High SAT
Low SAT
Guided
Exploratory
Learning Env

Fig 9

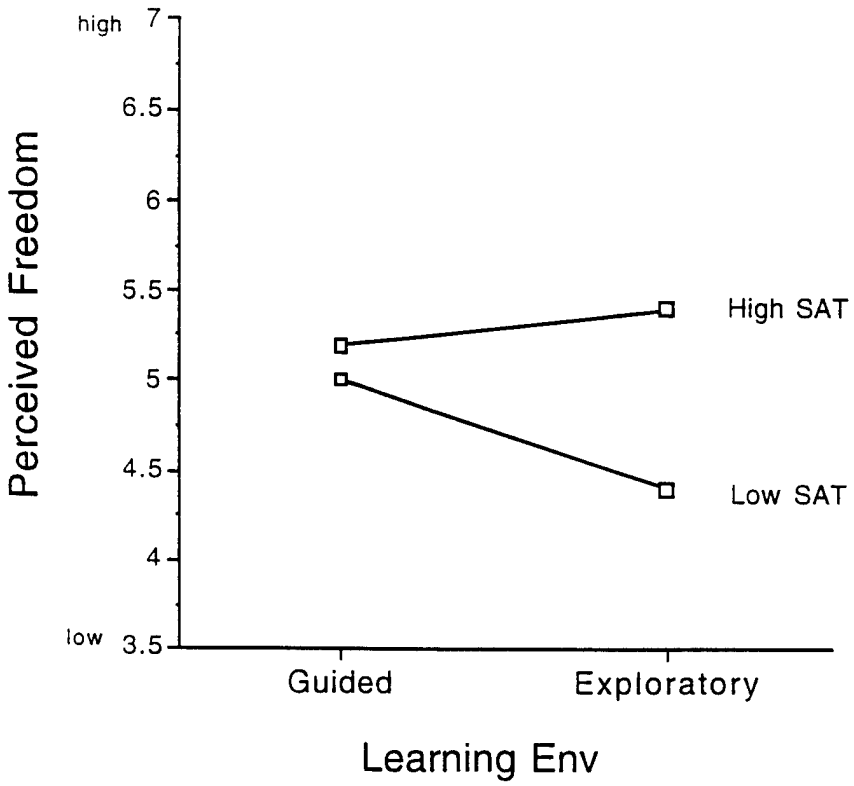
high
7
6.5
6
5.5
5
4.5
4
low
3.5
Perceived Freedom
High SAT
Low SAT
Guided
Exploratory
Learning Env

FIG 10

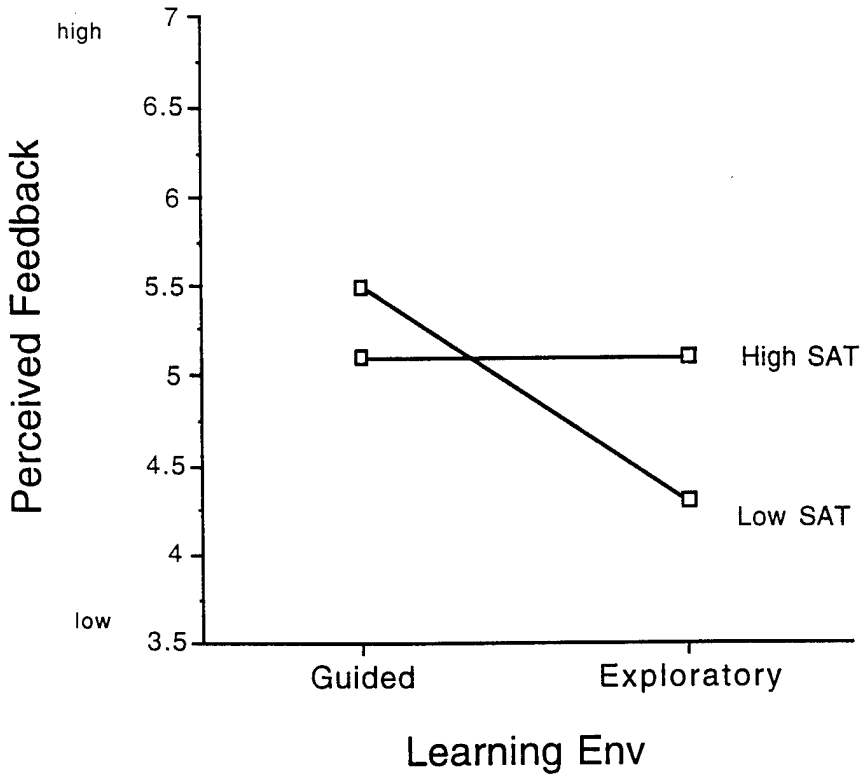
high
low
7
6.5
6
5.5
5
4.5
4
3.5
Perceived Feedback
High SAT
Low SAT
Guided
Exploratory
Learning Env

Fig 11